9 EQUITABLE: A Randomized Paging Algorithm

Here, we given a randomized online algorithm, EQUITABLE, for the k-paging problem, which is H_k-competitive against the oblivious adversary.

9.1 Offset Functions for the Paging Problem

The metric space of configurations for the k-paging problem is $\Lambda^k X$, where X (which could be huge) is the set of all pages in the system. Since no sensible algorithm would ever have two copies of one page in cache, we can ignore those configurations which have duplicates.

After any request sequence, the offset function is an integer function on $\Lambda^k X$ whose minimum is zero. But not all such functions could ever be offset functions of a request sequence.

Support. If ω is a work function (or offset function) for a metrical task system (M, d), we define the *Supp*(ω), the *support* of ω, to be the smallest set $S \subseteq M$ such that

$$\forall x \in M : \exists y \in S : \omega(x) = \omega(y) + d(x, y)$$

If M is finite, the support of ω is always well-defined. It is easy to construct work functions on infinite metrical task systems where the support of ω is not well-defined. However, if the metrical task system arises naturally, such as from the paging problem or the server problem, and ω is a reachable work function (or offset function) of the system, *i.e.,*is the work function or offset function of an actual request sequence, the support of ω is usually well-defined.

Theorem 1 *Let ω be the offset function of a request sequence for the k-paging problem. Then*
(a) *Supp*(ω) *is finite,*
(b) $\omega(x) = 0$ *for all* $x \in$ *Supp*(ω).

By Theorem 1, any offset function ω for the k-paging problem can be characterized by its support, which is a non-empty set of configurations, where each configuration is a set of k pages. If $x \in$ *Supp*(ω), then $x \subseteq \{p_1, \ldots p_k, r^1, \ldots r^n\}$, where $\{p_1, \ldots p_k\}$ is the original cache, and $r^1, \ldots r^n$ is the request sequence up to the current step. Thus, *Supp*(ω) is finite, since it can contain no more than $\binom{k+n}{k}$ configurations.[1]

Bar Notation. Let Σ be the alphabet consisting of all names of pages, together with a vertical bar symbol. Then, the bar notation for the support of any offset function for the k-paging problem is a string w over Σ such that

1. w contains exactly k bars, and the last symbol is a bar.
2. There are at least i page names to the left of the i^{th} bar.

If w is a string which satisfies the conditions above, we need to describe the offset function ω whose name is w. We do that by describing $S =$ *Supp*(ω). If $x \in \Lambda^k X$, then $x \in S$ if and only if, for all $1 \leq i \leq k$, x contains at least i pages whose names are to the left of the i^{th} bar.

For example, suppose $k = 3$, and $w = a|bc|de|$, where a, b, c, d, e are pages. Then $S = \{abc, abd, abe, acd, ace\}$. (By an abuse of notation, we denote the configuration $\{a, b, c\}$ simply by abc.)

[1] Since the last requested page must be in the cache, the *Supp*(ω) could not have cardinality more than $\binom{k-1+n}{k-1}$.

Alternative Notation. There can be more than one string which names the same offset function in the bar notation. We say those strings are *equivalent*. For example, $ab||$, $ba||$, $a|b|$, and $b|a|$ all name the same offset function, the one whose support is the singleton $\{ab\}$.

As a matter of style, I prefer to write $ab||$, while the originators of the bar notation prefer to write $a|b|$.

Exercise 9.1 *Write the support sets of the offset functions whose names are the following:*

1. $a|bcde|$
2. $ab||cd|$
3. $a|b|cd|$

Lemma 9.1 *Every reachable offset function for the k-paging problem can be expressed using the bar notation.*

The heart of the proof of Lemma 9.1 is the proof that if w is a bar notation string, and r is a new page, then the new offset function obtained by updating the current one can also be expressed using bar notation. We will use the notation $w \wedge r$ for the new bar notation string, where we read the symbol "$\wedge$" as "update."

Examples:

1. $ab|| \wedge c = c|ab|$ Fault.
2. $a|bc| \wedge b = ab||$ Hit.
3. $a|bc| \wedge d = d|abc|$ Fault.

Exercise 9.2 *Compute the bar notation for each of the following. following:*

1. $a|bcd| \wedge d$
2. $ab||cd| \wedge c$
3. $a|bc|def| \wedge d$
4. $a|bc|def| \wedge g$

9.2 The Distribution of EQUITABLE

EQUITABLE is a *stable distribution* algorithm. This means that its current distribution (which is all the adversary knows) is determined by the current offset function.

The *support* of a distribution on a finite set is the smallest set whose probability is 1. The support of the distribution of EQUITABLE is always exactly the same as the support of the offset function. For example, if the offset function is $a|bc|$, then its support is the set $\{ab, ac\}$. In this case, the distribution of EQUITABLE is

$$\tfrac{1}{2}ab + \tfrac{1}{2}ac$$

meaning that the probability is $\frac{1}{2}$ that our cache is ab, $\frac{1}{2}$ that it is ac, and zero that it is anything else.

The probability that a given configuration will be EQUITABLE's, given the bar notation for an offset function, is the outcome of the following Markov process. Let w be the bar notation string, and let S be the support of w. Let P be the set of all pages whose names appear in w. Let us say that the cardinality of P is n.

1. Choose a permutation of P, $(p_1, \ldots p_n)$ uniformly at random.
2. Let $x = \emptyset$.
3. For all i from 1 to n, in order: if $x \cup \{p_i\}$ is a subset of some member of S, then $x \leftarrow x \cup \{p_i\}$. Otherwise, do nothing.
4. Choose x to be the configuration.

Exercise 9.3 *Compute the distributions of EQUITABLE for the following offset functions.*

1. $ab||cd|$
2. $a|bcde|$
3. $ab||cde|$
4. $a||bc|de|$
5. $a|bcd||$
6. $a|bc|d|$

9.3 Cost

If the current offset function is ω, and a page p is requested, then we need to compute both the amortized optimal cost and the expected cost of EQUITABLE.

Lemma 9.2 *Let ω be the current offset function, and p the next request. Then the amortized optimal cost is zero if there is some $x \in Supp(\omega)$ such that $p \in x$, and one otherwise.*

Proof:(Sketch.) The support of $\omega \wedge p$ is precisely $\{x \in Supp(\omega) : p \in x\}$, provided that set is not empty. In that case, the amortized case is zero, because the optimal algorithm could have been at some configuration in $Supp(\omega \wedge p)$.

We can assume that the optimal algorithm is at some point in $Supp(\omega)$ before the request. If $p \notin x$ for all $x \in Supp(\omega)$, then the optimal algorithm must have a fault, and thus pays one. □

Lemma 9.3 *Let ω be the current offset function, and p the next request. Let $\pi_0 = \pi(\omega)$ be the distribution of EQUITABLE before the request, and $\pi_1 = \pi(\omega \wedge p)$ the distribution of EQUITABLE after the request. Then the expected cost of EQUITABLE for that step is the minimum transport distance between π_0 and π_1.*

Example Computation of EQUITABLE's Cost. Let $k = 3$, $\omega = a|bcd||$, and $p = b$. Then $\omega \wedge b = ab||cd|$. We can express the distributions before and after the request by the following table.

	abc	abd	acd
π_0	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$
π_1	$\frac{1}{2}$	$\frac{1}{2}$	0

The solution to the minimum transportation problem in this instance is to leave $\frac{1}{3}$ at abc, leave $\frac{1}{3}$ at abc, move $\frac{1}{6}$ from acd to abc, and $\frac{1}{6}$ from acd to abd. Each distance moved is 1, and thus the expected cost is $\frac{1}{3}$.

Exercise 9.4 *Compute the expected cost EQUITABLE in each of the following cases.*

1. $\omega = ab||$ *and* c *is requested.*
2. $\omega = a|bcd|$ *and* c *is requested.*
3. $\omega = a|bc|de|$ *and* c *is requested.*

9.4 Competitiveness of EQUITABLE

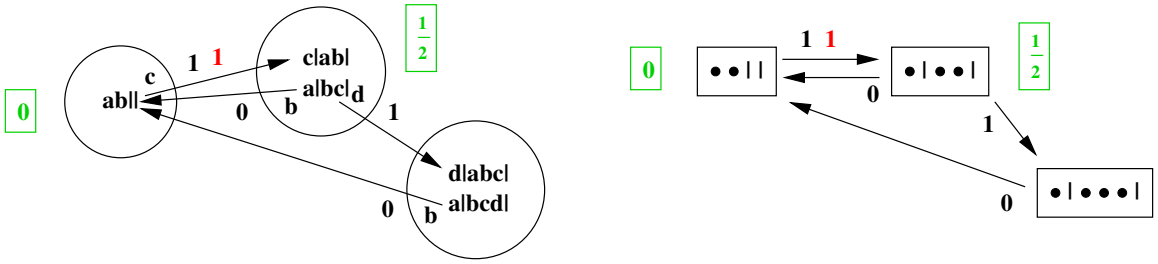


Figure 9.1: Game diagram for EQUITABLE for $k = 2$ (incomplete). The black numbers are amortized optimal costs. The red numbers are expected costs of EQUITABLE. The gree numbers are potentials.

9.5 Forgiveness in EQUITABLE

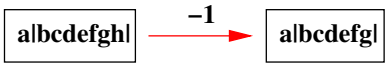


Figure 9.1: A forgiveness move of EQUITABLE.