A Better Lower Bound on the Competitive Ratio of the Randomized 2-Server Problem

Marek Chrobak * Lawrence L. Larmore, † Carsten Lund ‡ Nick Reingold §

Abstract

We present a lower bound of $1+e^{-1/2} \approx 1.6065$ on the competitive ratio of randomized algorithms for the weighted 2-cache problem, which is a special case of the 2-server problem. This improves the previously best known lower bound of $e/(e-1) \approx 1.582$ for both problems.

1 Introduction

The k-server problem is defined as follows: we are given k mobile servers that reside in a metric space M. At every time step, a request $r \in M$ is read, and in order to satisfy the request, we must move one server to the request point. Our cost is the total distance traveled by the servers. The choice of server at each step must be made online, *i.e.*, without knowledge of future requests, and therefore (except for some degenerate situations) no algorithm for scheduling the server movement can achieve the optimal cost on each request sequence. An online algorithm $\mathcal{A}$ is said to be *C-competitive* if there is a constant a, dependent only on the initial configuration, such that the cost incurred by $\mathcal{A}$ to service each request sequence ϱ is at most C times the optimal service cost for ϱ, plus a. Thus the ratio of $\mathcal{A}$'s cost and the optimal cost is, asymptotically, at most C. The goal is to design online k-server algorithms with small competitive ratios.

The k-server problem was introduced by Manasse, McGeoch and Sleator [10]. They show that k is a lower bound on the competitive ratio of any deterministic online algorithm for the k-server problem in any metric space with at least $k+1$ points. They also give an algorithm for the 2-server problem which is 2-competitive, and thus optimal, for any metric space. (See also [4].) For arbitrary $k \geq 3$, the best currently known upper bound on the competitive ratio is $2k-1$, as proven by Koutsoupias and Papadimitriou [7]. The upper bound of k is established in a number of special cases, including trees and spaces with at most $k+2$ points [3, 5, 8].

Randomization has proven to be very useful for online algorithms. For many online problems it has been shown that by using randomization one can achieve a dramatic improvement of the competitive ratio. For example, in a uniform space, *i.e.*, a space where all distances are equal, a randomized H_k-competitive algorithm for the k-server problem is known, where $H_k \approx \ln k$ is the k-th harmonic number [11, 1]. Metric spaces with three points have been investigated by [6] and [9].

The best currently known lower bound on the competitive ratio of a randomized 2-server online algorithm is $e/(e-1) \approx 1.582$. However, no randomized online algorithm is known for the 2-server problem which is better than 2-competitive for arbitrary metric spaces, nor, in fact, even for arbitrary

*Department of Computer Science, University of California Riverside, CA 92521. Email: `marek@cs.ucr.edu`. Research supported by NSF grant CCR-9503498.

†Department of Computer Science, University of Nevada, Las Vegas, NV 89154-4019. Email: `larmore@cs.unlv.edu`. Research supported by NSF grant CCR-9503441. Research partly done while visiting Institut Informatik V, Universität Bonn, and MPI für Informatik, Saarbrücken.

‡AT&T Labs, 600 Mountain Ave. Murray Hill, NJ 07940. Email: `lund@research.att.com`.

§AT&T Labs, 600 Mountain Ave. Murray Hill, NJ 07940. Email: `reingold@research.att.com`.

spaces with four points. For larger values of k, Blum *et al.* [2] give an asymptotic $\Omega(\sqrt{\log k/\log\log k})$ lower bound for an arbitrary space with at least $k+1$ points, and a $\Omega(\log k/\log\log k)$ lower bound for $k+1$ equally spaced points on the line.

In the *weighted-cache problem*, proposed by Manasse *et al.* [10], we are given a number of items, each item x having an associated weight, $w(x)$. Requests to items arrive online. Items can be stored in a cache that holds up to k items. When an item x is in the cache, the request to x can be satisfied at no cost. If x is not in the cache, it must brought into the cache at cost $w(x)$, replacing some item currently in the cache. A simple balancing algorithm for this problem with competitive ratio k was given in [3].

In the symmetric version of the weighted cache problem, the cost of replacing item x by y is defined to be $\frac{1}{2}(w(x)+w(y))$. Both versions, symmetric and asymmetric, can be reduced to each other, since any schedule for one version can be used for the other, increasing the cost by at most an additive constant. The symmetric version, in turn, is equivalent to the k-server problem in the star-shaped metric space defined as follows: the distance between items x, y is $xy = w(x) + w(y)$. One can think of the items covered by the k servers as the k items that are in the cache. Therefore the k-server algorithm for trees [5] is also k-competitive for the weighted cache.

In this paper we concentrate on randomized algorithms for the server problem and weighted cache, for $k = 2$. It has been conjectured that both problems have an $e/(e-1)$-competitive randomized algorithm. We prove that this is not the case, by providing a lower bound of $1 + e^{-1/2} \approx 1.6065$ on the competitive ratio of randomized algorithms for both problems.

2 Preliminaries

Online algorithms. We assume that an instance of the 2-server problem consists of a given metric space M and an initial server configuration X_0. We define an online 2-server algorithm to be a function $\mathcal{A}(\varrho)$ that to a given request sequence $\varrho \in M^*$ assigns the configuration after serving ϱ. In order to ensure that the requests are satisfied, we require that $r \in \mathcal{A}(\varrho)$, where r is the last request in ϱ. The cost of $\mathcal{A}$ on ϱ, $cost_{\mathcal{A}}(\varrho)$, is defined as the distance traveled by its servers while serving ϱ.

Competitiveness. Denote by $opt(\varrho)$ the optimal cost of serving the request sequence ϱ, *i.e.*, the cost paid by an optimal offline algorithm. After [10], define $\mathcal{A}$ to be C-*competitive* if there is a constant a such that for every request sequence ϱ

$$cost_{\mathcal{A}}(\varrho) \quad \leq \quad C \cdot opt(\varrho) + a. \tag{1}$$

The *competitive ratio* of $\mathcal{A}$ is defined to be the minimum C for which $\mathcal{A}$ is C-competitive.

Randomized algorithms. A randomized algorithm is defined as a probability distribution f on the set of all deterministic algorithms. We use notation $Exp_f[x]$ to denote the expected value of a random variable x under distribution f. Then the definition of competitiveness is the same as that for deterministic algorithms, (1), except that we need to replace $cost_{\mathcal{A}}(\varrho)$ by $Exp_f[cost_{\mathcal{A}}(\varrho)]$, the expected online algorithm's cost under distribution f.

The 2-server problem can be thought of as a game between two players: one of which is our online algorithm, and the other is an adversary which generates the request sequence and serves the requests with its own servers. In the randomized case there are several ways to define the rules of this game, depending on how much information the adversary is allowed to have when generating and serving the requests. The model we described above corresponds to the so-called *oblivious adversary*, which knows the description of our algorithm, but does not have access to its random choices in the course of the game.

The minimax principle. Yao [12] showed how to adapt the minimax principle from the theory of two-person zero-sum games to lower bound proofs in algorithm theory. His theorem is often referred to as Yao's Principle. We use an analogous approach to prove a lower bound on the competitive constant of randomized algorithms. The lemma below is probably not new (although we could not locate it in the literature), but we include the proof for completeness.

Lemma 1 *Suppose that for arbitrarily large D there is a probability distribution f on request sequences ϱ such that*

(a) $Exp_f[opt(\varrho)] \geq D$, and

(b) $Exp_f[\mathrm{cost}_{\mathcal{A}}(\varrho)] \geq C \cdot Exp_f[opt(\varrho)]$ for each deterministic online algorithm $\mathcal{A}$.

Then no randomized online algorithm is C'-competitive for any $C' < C$.

Proof: Towards contradiction, suppose that $\mathcal{B}$ is a C'-competitive randomized online algorithm, for $C' < C$. Recall that $\mathcal{B}$ is a distribution on the set of all deterministic algorithms $\mathcal{A}$. Pick a constant a such that $Exp_{\mathcal{B}}[cost_{\mathcal{A}}(\varrho)] \leq C' \cdot opt(\varrho) + a$ for every request sequence ϱ. Choose $D > a/(C - C')$. Then

$$\begin{aligned} Exp_f Exp_{\mathcal{B}}[cost_{\mathcal{A}}(\varrho)] &= Exp_{\mathcal{B}} Exp_f[cost_{\mathcal{A}}(\varrho)] \\ &\geq Exp_{\mathcal{B}}[C \cdot Exp_f[opt(\varrho)]] \\ &= C \cdot Exp_f[opt(\varrho)] \\ &> C' \cdot Exp_f[opt(\varrho)] + a \end{aligned}$$

– a contradiction. □

3 The $1 + e^{-1/2}$ Lower Bound

Let $C_0 = 1 + e^{-1/2}$. Our proof of the C_0 lower bound is based on Lemma 1. We present a metric space M, and an adversary strategy $\mathcal{G}$ in M. $\mathcal{G}$ is a probability distribution on request sequences for which the expected optimal cost is arbitrarily high, and the expected cost of each deterministic online algorithm will be at least $C_0 - \delta$ times the expected optimal cost, where δ can be made arbitrarily small.

Let $n, m \geq 1$ be large integers, whose values will be determined later. Let $\alpha = 1/(2n)$. Define a metric space $M = \{a, b, c, z\}$ to be the star with center point z, and with points a, b, and c at the ends of arms of length α, 1, and 1, respectively. Thus, $az = \alpha$, $bz = cz = 1$, $ab = ac = 1 + \alpha$, and $bc = 2$. The initial configuration, *i.e.*, the set of positions of our two servers, is $\{b, z\}$.

The adversary strategy. We define $\mathcal{G}$ by a random process that generates a request sequence. For simplicity, we will use the same notation $\mathcal{G}$ to denote this process. $\mathcal{G}$ always chooses the request sequence to be the concatenation of pairs of the form xz, where $x \in \{a, b, c\}$. $\mathcal{G}$ consists of m phases. We say that $\mathcal{G}$ *is in state* $\{x, z\}$ if $\mathcal{G}$'s two servers are at x and z. Before and after each phase, $\mathcal{G}$ is in one of two states, $\{b, z\}$ or $\{c, z\}$. Initially, $\mathcal{G}$ is in state $\{b, z\}$.

We now describe one phase. By symmetry, we can assume $\mathcal{G}$ is in state $\{b, z\}$ at the beginning of that phase. Then $\mathcal{G}$ makes the following random choice of a request sequence ϱ:

(bb0) $\varrho = (az)^i (bz)^{2m+1}$ with probability $\alpha(1-\alpha)^{i-1}$ for each $i = 1, \ldots, n-1$.

(bb1) $\varrho = (az)^n (czbz)^j (bz)^{2m+1}$ with probability $2^{-2j}(1-\alpha)^{n-1}$ for $j \geq 1$.

(bc1) $\varrho = (az)^n (czbz)^j (cz)^{2m+2}$ with probability $2^{-2j-1}(1-\alpha)^{n-1}$, for $j \geq 0$.

If type (bb0) or (bb1) sequences are chosen, $\mathcal{G}$ completes the phase in state $\{b, z\}$; if type (bc1) is chosen, $\mathcal{G}$ completes the phase in state $\{c, z\}$. (Of course, if a phase starts with $\mathcal{G}$ in state $\{c, z\}$, there are corresponding types (cc0), (cc1) and (cb1), defined symmetrically.)

Informally, one can think of a phase as consisting of three subphases. First, $\mathcal{G}$ requests az. Then, $\mathcal{G}$ repeatedly requests az with probability $(1 - \alpha)$ and bz with probability α until one of two things happens: either bz is requested once or az is requested a total of n times (including the first request). This ends the first subphase.

If the first subphase ends with bz, the second subphase is empty. Otherwise, the second subphase begins with cz. Subsequently, $\mathcal{G}$ repeatedly requests bz with probability $\frac{1}{2}$ and cz with probability $\frac{1}{2}$, until either bz or cz is requested twice consecutively. This ends the second subphase.

In the third subphase, $\mathcal{G}$ requests $(xz)^{2m}$, for $x \in \{b, c\}$, where xz is the pair that was requested last in the previous subphase. $\mathcal{G}$ ends the phase in state $\{x, z\}$.

Lemma 2 *For each $\epsilon > 0$, if n is large enough, the expected value of the optimal cost in each phase is at most $2 + \epsilon$.*

Proof: Fix $\epsilon > 0$. We present an adversary schedule whose configurations before and after each phase are the same as the states of $\mathcal{G}$, and whose cost is at most $2 + \epsilon$ in each phase if n is large enough.

Suppose a given phase starts in state $\{b, z\}$. The adversary has its servers at $\{b, z\}$ as well. Call these servers s_b and s_z. If the final state for that phase is $\{b, z\}$, then the phase can be served by leaving s_b at b and servicing all requests other than b with s_z. If the final state is $\{c, z\}$, then the phase can be served as follows: serve the first request on a with s_z, serve the first request on z with s_b, and then, when c is requested, move s_b to c and leave it there, servicing all requests at other points with s_z. The costs of this schedule in the three cases (bb0), (bb1), (bc1) are $2i\alpha$, $2n\alpha + 2j$ and $2\alpha + 2 + 2j$, respectively. Let Δopt denote the random variable equal to the optimal cost of this phase. Then the expected value of Δopt is

$$\begin{aligned} Exp_{\mathcal{G}}[\Delta opt] &\le \sum_{i=1}^{n-1} (2i\alpha)\alpha(1-\alpha)^{i-1} + \sum_{j=1}^{\infty} (2n\alpha + 2j)2^{-2j}(1-\alpha)^{n-1} \\ &\quad + \sum_{j=0}^{\infty} (2\alpha + 2 + 2j)2^{-2j-1}(1-\alpha)^{n-1} \\ &= 2 + \frac{10}{3}\alpha(1-\alpha)^{n-1}, \end{aligned}$$

which converges to 2 as n approaches ∞. $\square$

Without loss of generality, we can assume that a given online algorithm always keeps its servers at two different points and that it only moves a server to the request point. Furthermore, by the structure of the metric space and the form of request sequences generated by $\mathcal{G}$, we can also assume that all requests on a, b, c are served with the server on that point, if there is one, and by moving the server from z, otherwise. We will call such algorithms *reasonable*. Any algorithm can be converted into a reasonable one without increasing cost. Therefore we can also assume that a randomized algorithm is reasonable with probability one.

Lemma 3 *Let $\mathcal{A}$ be any reasonable deterministic algorithm whose configuration after each phase is the same as $\mathcal{G}$'s state. For each $\epsilon > 0$, if n is large enough, the expected cost of $\mathcal{A}$ in each phase is at least $2 + 2e^{-1/2} - \epsilon$.*

Proof: Choose the maximum $0 \le k \le n$ such that $\mathcal{A}$ is in state $\{b, z\}$ after serving $(az)^k$. If $k < n$, then $\mathcal{A}$ is in state $\{a, z\}$ after serving $(az)^i$, for any $k < i \le n$. After serving $(az)^n cz$, and for the remainder of the phase, we may assume that $\mathcal{A}$ has no server at a, since it is pointless for $\mathcal{A}$ to have a server at a, where there will be no further requests during that phase.

Let $\Delta cost_{\mathcal{A}}$ be a random variable denoting the cost of $\mathcal{A}$ in this phase. We compute $\Delta cost_{\mathcal{A}}$ separately for the cases when $k = n$ and $k < n$.

Suppose first that $k = n$. Then $\mathcal{A}$ pays $2i\alpha$ on a sequence of type (bb0). On sequences of types (bb1), (bc1), $\mathcal{A}$ pays $2n\alpha$ in the first subphase, and then it pays 2 to service cz. At any time when the second subphase has already started, *i.e.*, the adversary has already requested cz at least once, but the second subphase has not ended, let E denote the expected cost of $\mathcal{A}$ in the remainder of the second subphase. The expected cost of $\mathcal{A}$ at the next request pair is 1. Thus E satisfies the recurrence $E = \frac{1}{2}(2 + E)$, implying that $E = 2$. Summarizing, we conclude that the expected cost of $\mathcal{A}$ on sequences (bb1) and (bc1) is $2\alpha n + 2 + E = 2\alpha n + 4$. Therefore, for $k = n$, we obtain

$$\begin{aligned} Exp_{\mathcal{G}}[\Delta cost_{\mathcal{A}}] &= \sum_{i=1}^{n-1}(2i\alpha)\alpha(1-\alpha)^{i-1} + (2\alpha n + 4)(1-\alpha)^{n-1} \\ &= 2 + 2(1-\alpha)^{n-1} + 2\alpha(1-\alpha)^{n-1}, \end{aligned}$$

which converges to $2 + 2e^{-1/2}$ as n approaches ∞.

Consider now the case when $k < n$. For a sequence ϱ of type (bb0) and $i = 1, \ldots, k$, the cost of $\mathcal{A}$ is $2i\alpha$. For $i = k+1, \ldots, n-1$, $\mathcal{A}$'s cost is $2k\alpha + \alpha + 1$ for serving $(az)^i$, plus $1 + \alpha$ to serve the rest of ϱ, for the total of $2k\alpha + 2\alpha + 2$. By an argument similar to the previous case, the expected cost of $\mathcal{A}$ when ϱ is chosen from among sequences (bb1) or (bc1) is $2k\alpha + 2\alpha + 2 + E = 2k\alpha + 2\alpha + 4$. Therefore, for $k < n$, we obtain

$$\begin{aligned} Exp_{\mathcal{G}}[\Delta cost_{\mathcal{A}}] &= \sum_{i=1}^{k}(2i\alpha)\alpha(1-\alpha)^{i-1} + \sum_{i=k+1}^{n-1}(2k\alpha + 2\alpha + 2)\alpha(1-\alpha)^{i-1} \\ &\quad + (1-\alpha)^{n-1}(2k\alpha + 2\alpha + 4) \\ &= 2 + 2(1-\alpha)^{n-1} + 2\alpha(1-\alpha)^{k}, \end{aligned}$$

which converges to $2 + 2e^{-1/2}$ as n approaches ∞. $\Box$

Theorem 1 *No randomized algorithm for the weighted 2-cache problem (and thus also for the 2-server problem) is better than C_0-competitive, where $C_0 = 1 + e^{-1/2}$.*

Proof: Consider an online algorithm $\mathcal{A}$. Without loss of generality, we can assume that $\mathcal{A}$ is reasonable. If $\mathcal{A}$ satisfies the assumption of Lemma 3, the ratio of its cost to the optimal cost in each phase is arbitrarily close to C_0, by Lemmas 2 and 3. Otherwise, we pick the first phase after which the configuration of $\mathcal{A}$ differs from $\mathcal{G}$'s state. Suppose that at the completion of this phase, $\mathcal{G}$ is in state $\{b, z\}$ while $\mathcal{A}$ is in configuration $\{c, z\}$. Then $\mathcal{A}$ must have been serving the suffix $(bz)^{2m}$ with one server, at cost at least $4m$. Therefore, by Lemma 2, if n is large enough then the ratio of $\mathcal{A}$'s cost to the optimal cost, for the whole request sequence, is greater than C_0.

We conclude that for each $C < C_0$ we can pick n, m for which the adversary strategy $\mathcal{G}$ satisfies the assumptions of Lemma 1. Therefore no randomized online algorithm is better than C-competitive. This implies the theorem. $\Box$

References

[1] D. Achlioptas, M. Chrobak, and J. Noga. Competitive analysis of randomized paging algorithms. In *4th Annual European Symposium on Algorithms*, pages 419–430, 1996.

[2] A. Blum, H. Karloff, Y. Rabani, and M. Saks. A decomposition theorem and lower bounds for randomized server problems. In *Proc. 33rd IEEE Symposium on Foundations of Computer Science*, pages 197–207, 1992.

[3] M. Chrobak, H. Karloff, T. H. Payne, and S. Vishwanathan. New results on server problems. *SIAM Journal on Discrete Mathematics*, 4:172–181, 1991. Also in Proceedings of the 1st Annual ACM-SIAM Symposium on Discrete Algorithms, San Francisco, 1990, pp. 291-300.

[4] M. Chrobak and L. L. Larmore. A new approach to the server problem. *SIAM Journal on Discrete Mathematics*, 4:323–328, 1991.

[5] M. Chrobak and L. L. Larmore. An optimal online algorithm for k servers on trees. *SIAM Journal on Computing*, 20:144–148, 1991.

[6] A. Karlin, M. Manasse, L. McGeoch, and S. Owicki. Randomized competitive algorithms for non-uniform problems. In *1st Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 301–309, 1990. To appear in *Algorithmica*.

[7] E. Koutsoupias and C. Papadimitriou. On the k-server conjecture. In *Proc. 25th Symposium on Theory of Computing*, pages 507–511, 1994.

[8] E. Koutsoupias and C. Papadimitriou. The 2-evader problem. *Information Processing Letters*, 57:249–252, 1996.

[9] C. Lund and N. Reingold. Linear programs for randomized on-line algorithms. In *Proc. 5th ACM-SIAM Symposium on Discrete Algorithms*, 1994. 382–391.

[10] M. Manasse, L. A. McGeoch, and D. Sleator. Competitive algorithms for server problems. *Journal of Algorithms*, 11:208–230, 1990.

[11] L. McGeoch and D. Sleator. A strongly competitive randomized paging algorithm. *J. Algorithms*, 6:816–825, 1991.

[12] A. C. Yao. Probabilistic computations: Towards a unified measure of complexity. In *Proc. 12th ACM Symposium on Theory of Computing*, 1980.