On Fast Algorithms for Two Servers

Marek Chrobak Lawrence L. Larmore
Department of Computer Science
University of California, Riverside, CA 92521

October 4, 1996

Abstract

We consider 2-server algorithms with time complexity $O(1)$ per each request. We show that the previously known algorithm BALANCE2 has competitiveness constant not better than 6, and present another algorithm whose competitiveness constant is 4.

1 Introduction

The k-server problem can be formulated as follows: Let M be a metric space, in which we have k mobile servers that can occupy points of M. At each time step we are given a request, specified by a location $r \in M$, and we have to choose which server to move to r to "serve" the request. Our measure of cost is the distance traveled by our servers, and the task is to design algorithms that minimize that cost. The requests have to be served on-line, that is, the choice of the server at the current step cannot depend on the future requests.

Let $\mathcal{A}$ be a server algorithm. Given an initial configuration $K_0 \in M^k$ of our k servers and a sequence of requests $R \in M^*$, let $cost_{\mathcal{A}}(K_0, R)$ be the total cost of serving the requests of R by $\mathcal{A}$ when the servers start from K_0, and $cost_{opt}(K_0, R)$ the corresponding cost of the optimal schedule. Then $\mathcal{A}$ is called *c-competitive* if for any sequence R of requests we have

$$cost_{\mathcal{A}}(K_0, R) \le c \cdot cost_{opt}(K_0, R) + b,$$

for some constant b that may depend only on the initial configuration K_0. Manasse, McGeoch and Sleator [8] proved that there are no k-server c-competitive algorithms for $c < k$ in any metric space with at least $k+1$ points. The problem of whether there is an on-line algorithm that achieves $c = k$ for each k, is open. The famous "k-server conjecture" from [8] states that this is indeed true. Up to now, it has been proven only for $k = 2$ in [8]. Another 2-competitive algorithm for 2 servers was given in [4].

The harmonic algorithm, referred to later as HARMONIC, moves a server to the request with probability proportional to the inverse of its distance to the request point. HARMONIC is an example of a *memoryless* algorithm, whose decision depends only on the current configuration of the servers. Raghavan and Snir [9] proved that HARMONIC has competitiveness constant between 3 and 6 for two servers. For $k = 3$, Berman *et al* [1] proved that HARMONIC is competitive, but its competitive constant was not given.

For general k, the solution is known only for some specific metric spaces. In [3] a k-competitive algorithm is given for trees. Coppersmith *et al* [5] gave a randomized k-competitive algorithm for a broad class of metric spaces, that also includes trees.

The known 2-competitive deterministic algorithms for two servers from [4, 8] are very space- and time-consuming. If the given metric space has m points, then the algorithms need $O(m)$ space and $O(m)$ time per each request. Thus on a sequence of n requests in an arbitrary metric space, the space needed is $O(n)$ and the total time is $O(n^2)$. Clearly, no server system can allow such delays in practice.

This has motivated research to find faster competitive algorithms for two servers. Notice that HARMONIC balances, on average, the work of each server. This suggests that perhaps a *deterministic* algorithm that balances the work of the servers is also competitive. Denote by $cost_i$ the current work of server s_i. Algorithm BALANCE chooses, given a request at r, the server s_i that minimizes $cost_i + |s_i r|$ (we denote the distance between $x, y \in M$ by $|xy|$ and for simplicity use s_i to denote both the i-th server and its current location). In [8] it was proven that this algorithm is k-competitive if the underlying metric space has only $k+1$ points, but it is not hard to show that even for two servers and four points it is not competitive at all [7]. Another proposed algorithm, called here BALANCE2, always chooses the server that minimizes $cost_i + 2 \cdot |s_i r|$. Irani and Rubinfeld [6] proved that BALANCE2 is 10-competitive for two servers, and it has been known that its competitiveness constant is not better than 4. Even though its competitiveness constant is worse than other known algorithms, the advantage of this approach is that BALANCE2 needs only one real variable: $cost_1 - cost_2$, and it is time-optimal since each step needs only $O(1)$ time.

In this paper we improve those results. First, in Section 3, we show the competitiveness constant of BALANCE2 is not better than 6. In Section 4 we present the main result of the paper. We give a deterministic algorithm BALANCE_SLACK that balances the accumulated slack of the servers (as defined later), and show that it is 4-competitive, and that the constant 4 is tight. This algorithm is also 2-competitive, and thus optimal, when the underlying metric space is a tree.

2 The Proof Techniques

We picture the behavior of a server algorithm $\mathcal{A}$ as a game between $\mathcal{A}$'s servers s_1, s_2 and an adversary's servers a_1 and a_2. Initially the adversary's servers are at the same positions as our servers. The adversary is allowed to create requests and serve them with its own servers trying to force the ratio $cost_{adv}/cost_{\mathcal{A}}$ be as large as possible. In the proofs, we show that for any initial configuration K_0 and each sequence of requests R, independently of the adversary's behavior, the inequality

$$cost_{\mathcal{A}}(K_0, R) \le c \cdot cost_{adv}(K_0, R) + b,$$

always holds. (In the rest of the paper we will omit the parameters K_0 and R when they are understood.) This implies that the algorithm is c-competitive, since one of the possible adversary's computation will correspond to the optimal schedule on R.

The proofs are based on the so-called potential technique, used often in the analysis of dynamic data structures. We define a function Φ, called a *potential*, that maps all possible configurations of the servers (ours and adversary's) into the non-negative reals. (A configuration includes the state of the memory of the algorithm.)

The game consists of a number of rounds, each consisting of one adversary move and one algorithm move. The adversary moves a single server and puts the request on the position of this server. Then, one of our servers, designated by the algorithm, moves to the request point. It is not hard to see that these rules lead to no loss of generality. Our proof is based on the following lemma which can be proven by simple summation over the whole sequence of requests.

Lemma 1 *Let $\Delta\Phi$, $\Delta cost_{adv}$ and $\Delta cost_{\mathcal{A}}$ denote, respectively, the changes of the potential, the adversary's cost and our cost in the current round. Suppose that*

$$\Delta\Phi + \Delta cost_{\mathcal{A}} \le c \cdot \Delta cost_{adv},$$

in every round. Then $\mathcal{A}$ is c-competitive.

3 On the Competitiveness Constant of BALANCE2

We prove now that the competitiveness constant of BALANCE2 is at least 6. Let M be the metric space from Figure 1. We will show that in this space the adversary has a strategy in which BALANCE2's cost is asymptotically at least six times his cost. When our algorithm has a choice of moves, that is when $cost_1 + 2 \cdot |s_1 r| = cost_2 + 2 \cdot |s_2 r|$, the adversary will dictate which server is to be moved. This can be done because the adversary can remove this ambiguity by perturbing the distances very slightly.

Initially, $s_1 = a_1 = x_1$, $s_2 = x_3$, $a_2 = y_2$ and $cost_2 - cost_1 = 1$. Next, the adversary puts the request on a_2 and it will be served by s_1. Now $cost_2 - cost_1 = -6$. The adversary moves a_2 to y_3 and puts the request there. Now $cost_i + 2 \cdot |s_i y_3|$ are equal for $i = 1, 2$. The adversary makes s_1 serve the request. His cost in these two moves is 2, and BALANCE2's cost is 12. At the end of those two rounds, $s_2 = a_2 = y_3$, $s_1 = y_2$, $a_1 = x_1$, and $cost_1 - cost_2 = 1$. But this new configuration is symmetric to the one we started from, and the adversary can continue these moves forever.

Thus, by the result from [6], we obtain the following:

Theorem 1 *The competitiveness constant of BALANCE2 is in the interval* $[6, 10]$.

Insert Figure 1 here

4 The Algorithm

In order to give the reader some more intuition, let us consider first a special configuration of our servers s_1, s_2 and the request point r where s_2 is between s_1 and r, that is $|s_1 s_2| + |s_2 r| = |s_1 r|$. It is not hard to see that for every algorithm $\mathcal{A}$ there is another algorithm $\mathcal{A}'$, with the same or better constant, such that in this configuration $\mathcal{A}'$ will always move s_2. Note that the cost of moving s_1 to r is the same as the cost of moving s_1 to s_2 and s_2 to r. Thus $\mathcal{A}'$ can move s_2 to r, and, instead of actually moving s_1 to s_2, just remember that the virtual position of s_1 is on the previous location of s_2, and later, when s_1 serves the request, move s_1 to this new request point. By the triangle inequality, the cost of $\mathcal{A}'$ will not exceed the cost of $\mathcal{A}$. Notice that if, for example, s_1 and s_2 are very close to each other in this configuration, HARMONIC will move them with about the same probability. This example gives insight why HARMONIC is not optimal. Even if we perturb the position of r slightly, it appears that we should move s_2 with much higher probability than s_1. This is one of the basic ideas leading to our algorithm.

Let M be a metric space. For $x, y, z \in M$, let $\mu_{x,y,z} = \mu(x,y,z) = \frac{1}{2} \cdot (|xy| + |xz| - |yz|)$. Notice that $\mu_{x,y,z} = \mu_{x,z,y}$, and $\mu_{x,y,z} + \mu_{y,x,z} = |xy|$. Given a configuration of our servers, and the request point r, by the *slack* of s_i we mean $\epsilon_i = \mu(s_i, s_j, r)$. Note that $\epsilon_1 + \epsilon_2 = |s_1 s_2|$.

Let us consider the following randomized algorithm:

Algorithm RANDOM_SLACK: Move s_i to the request point r with probability $p_i = \frac{\epsilon_j}{|s_1 s_2|}$, where $j \neq i$.

P.Raghavan pointed out to us that this algorithm is equivalent to the one given in [5], and therefore we obtain that RANDOM_SLACK is 2-competitive. Thus it appears that balancing the accumulated slack of a server, instead of the total work, should give a better algorithm than BALANCE2. Below we show that this is indeed true.

Algorithm BALANCE_SLACK: Let e_i denote the *cumultative slack* of server s_i, where e_i is initially zero, and it is updated at each step as follows. **If** the request is on r, **then choose** i such that $e_i + \epsilon_i$ is minimized, **move** s_i to r and **set** $e_i \leftarrow e_i + \epsilon_i$.

Theorem 2 *Algorithm BALANCE_SLACK is 4-competitive.*

Proof: Throughout this proof, we will eliminate the bars in the notation for the metric in

M. In other words, if $x, y \in M$, we write xy instead of $|xy|$. Between rounds and before the first round, one of our servers must be at the same point in M as one adversary server. By relabeling the servers, if necessary, we will always assume that $a_1 = s_1$. We define the potential

$$\Phi = s_1 s_2 + 2 s_2 a_2 + 2|e_2 - e_1 + \mu(s_2, s_1, a_2)|,$$

Note that the first two terms constitute the Coppersmith-Doyle-Raghavan-Snir potential [5], while the quantity inside the absolute value signs is what the difference in the accumulated slack will be if the next move by the adversary is to make a request at a_2 and if s_2 responds, resulting in a "matched" configuration. Also notice that when our and the adversary's servers are on the same two points then the value of Φ does not depend on which matched pair of servers will be labeled s_1 and a_1.

We show that, for each round, the following claim holds:

Claim: $\Delta\Phi + \Delta cost_{\mathcal{A}} \leq 4 \cdot \Delta cost_{adv}$.

Without loss of generality, $a_1 = s_1 = x, s_2 = y, a_2 = z$, and the request point is r, where $x, y, z, r \in M$. Write $\delta = e_2 - e_1$. Then $\Phi = xy + 2 \cdot yz + |2 \cdot \delta + xy - xz + yz|$.

The proof breaks into four cases, depending on which adversary server makes the request and which of our servers serves that request. All have similar proofs, using the triangle inequality for both M and the set of real numbers. We use the notation δ', Φ', to denote the value of δ and the potential after the round.

Case 1: a_1 requests, s_1 serves. In this case $\delta' = \delta - \epsilon_1 = \delta - \frac{1}{2} \cdot (xr + xy - yr)$, and $\Delta cost_{\mathcal{A}} = \Delta cost_{adv} = xr$. Then $\Phi' = yr + 2 \cdot yz + |2 \cdot \delta - xr - xy + 2 \cdot yr + yz - zr|$, and therefore

$$\begin{aligned}
\Delta\Phi + \Delta cost_{\mathcal{A}} &= xr - xy + yr + |2 \cdot \delta - xr - xy + 2 \cdot yr + yz - zr| - |2 \cdot \delta + xy - xz + yz| \\
&\leq xr - xy + yr + |xr + 2 \cdot xy - xz - 2 \cdot yr + zr|
\end{aligned}$$

We now split the computation into two branches. Subcase: the quantity inside the absolute value signs is non-negative:

$$= \quad 2 \cdot xr + (xy - yr) + (zr - xz) \leq 4 \cdot xr = 4 \cdot \Delta cost_{adv}.$$

Subcase: the quantity inside the absolute value signs is negative:

$$= \quad (3 \cdot yr - 3 \cdot xy) + (xz - zr) \leq 4 \cdot xr = 4 \cdot \Delta cost_{adv}.$$

Case 2: a_1 requests, s_2 serves. Then $\delta' = -(\delta + \epsilon_2) = -\delta - \frac{1}{2} \cdot (yr + xy - xr)$ (notice the change of sign because of the relabeling of our servers), so $\Phi' = xr + 2 \cdot xz + |2 \cdot \delta - 2 \cdot xr + xy - xz + yr + zr|$. Also $\Delta cost_{adv} = xr$, and $\Delta cost_{\mathcal{A}} = yr$. Since s_2 moves, $e_2 + \epsilon_2 \leq e_1 + \epsilon_1$, which implies that $xr - yr - \delta \geq 0$. Since $\xi + |\mu| \geq \xi - \mu$ for any ξ, μ, we have $\Phi = xy + 2 \cdot yz + |2 \cdot \delta + xy - xz + yz| \geq -2 \cdot \delta + yz + xz$. Therefore

$$\Delta\Phi + \Delta cost_{\mathcal{A}} \quad \leq \quad 2 \cdot \delta + xr + xz + yr - yz + |2 \cdot \delta - 2 \cdot xr + xy - xz + yr + zr|$$

We now split the computation into two branches. Subcase: the quantity inside the absolute value signs is non-negative:

$$\begin{aligned} &= 4\cdot\delta - xr + xy + 2\cdot yr - yz + zr \\ &\leq 4\cdot\delta - xr + xy + 2\cdot yr - yz + zr + 4\cdot(xr - yr - \delta) \\ &= 3\cdot xr + (xy - yr) + (zr - yz - yr) \ \leq\ 4\cdot xr \ =\ 4\cdot\Delta cost_{adv}. \end{aligned}$$

Subcase: the quantity inside the absolute value signs is negative:

$$= \ 3\cdot xr + (xz - zr) + (xz - xy - yz) \leq 4\cdot xr \ =\ 4\cdot\Delta cost_{adv}.$$

Case 3: a_2 requests, s_1 serves. In this case we have $\delta' = \delta - \frac{1}{2}\cdot(xr + xy - yr)$ and thus $\Phi' = yr + 2\cdot xy + |2\cdot\delta - 2\cdot xr + 2\cdot yr|$. Notice that the adversary's servers have been relabeled. Since s_1 moves, we have $\delta + yr - xr \geq 0$, hence the absolute value signs may be removed from the formula for Φ'. Also, $\Phi = xy + 2\cdot yz + |2\cdot\delta + xy - xz + yz| \geq 2\cdot xy + 3\cdot yz - xz + 2\cdot\delta$. $\Delta cost_{\mathcal{A}} = xr$, and $\Delta cost_{adv} = zr$. Therefore

$$\Delta\Phi + \Delta cost_{\mathcal{A}} \leq (3\cdot yr - 3\cdot yz) + (xz - xr) \ \leq\ 4\cdot zr \ =\ 4\cdot\Delta cost_{adv}.$$

Case 4: a_2 requests, s_2 serves. In this case we have $\delta' = \delta + \frac{1}{2}\cdot(xy - xr + yr)$. $\Phi' = xr + |2\cdot\delta + xy - xr + yr|$, $\Delta cost_{\mathcal{A}} = yr$, and $\Delta cost_{adv} = zr$. Therefore

$$\begin{aligned} \Delta\Phi + \Delta cost_{\mathcal{A}} &\leq xr - xy + yr - 2\cdot yz + |-xr + xz - yz + yr| \\ &\leq (xr - xy - yz) + (yr - yz) + |xz - xr| + |yr - yz| \\ &\leq 4\cdot zr \ =\ 4\cdot\Delta cost_{adv}. \end{aligned}$$

This completes the proof.□

The result of Theorem 2 is tight, as the following example shows.

Example showing competitiveness of BALANCE_SLACK cannot be less than 4. Let M be a circle of circumference 10, where the distance between points is the length of the shortest arc between them. We think of M as being the interval $[0, 10]$ with 0 and 10 identified.

Let K_0 be the configuration where a_1 and s_1 are on point 0, s_2 is on point 1, a_2 is on point 6, and $e_2 = e_1 - 1$. The adversary makes request at a_2, and s_1 moves to meet that request. The value of e_i do not change. Next, the adversary moves a_2 to point 5 and makes a request, which is served by s_1 again. The values of e_1 increases by 1. Next, adversary moves a_2 to point 4 and makes a request, which is served by s_2. The value of e_2 increases by 3. In this sequence of rounds, $cost_{\mathcal{A}}$ increased by 8, and $cost_{adv}$ increased by 2. The configuration is now symmetric to the original configuration K_0. By repeating this sequence indefinitely, the adversary can force the limiting ratio of algorithm cost to its cost to be 4.

We also can prove the following result.

Theorem 3 *If M is a tree then BALANCE_SLACK is 2-competitive on M.*

Proof: The proof is similar to the previous one, and we leave the details to the reader. We only show how to define the appropriate potential function.

For a server s_i and $x \in M$ define the *virtual distance* between s_i and x, denoted $d(s_i, x)$, as follows: if $e_i \geq e_j$, then $d(s_i, x) = |s_i x|$. Otherwise, $d(s_i, x) = \mu(x, s_1, s_2) + |\mu(s_i, s_j, x) + e_i - e_j|$. Let m be the weight of a minimum matching in the bipartite graph whose components are $\{s_1, s_2\}$ and $\{a_1, a_2\}$, but the distances between s_i and a_p are now measured as $d(s_i, a_p)$. Then the potential is $\Phi = 2 \cdot m + |s_1 s_2|$. $\Box$

We immediately obtain:

Corollary 1 *If M can be embedded isometrically into a tree, then BALANCE_SLACK is 2-competitive on M.*

References

[1] P.Berman, H.Karloff, G.Tardos, A competitive algorithm for three servers, in Proc. First Annual ACM-SIAM Symposium on Discrete Algorithms, 1990.

[2] M.Chrobak, H.Karloff, T.Payne, S.Vishwanathan, New results on server problems, in Proc. First Annual ACM-SIAM Symposium on Discrete Algorithms, 1990.

[3] M.Chrobak, L.Larmore, An optimal on-line algorithm for trees, to appear in SIAM J. Computing.

[4] M.Chrobak, L.Larmore, A new approach to the server problem, submitted.

[5] D.Coppersmith, P.G.Doyle, P.Raghavan, M.Snir, Random walks on weighted graphs and applications to on-line algorithms, manuscript.

[6] S.Irani and R.Rubinfeld, A competitive 2-server algorithm, manuscript.

[7] M.Manasse, private communication.

[8] M.Manasse, L.A.McGeoch, D.Sleator, Competitive algorithms for server problems, Proc. 20th ACM STOC (1988) 322-333.

[9] P.Raghavan, M.Snir, Memory versus randomization in on-line algorithms, Proc. ICALP 1989.

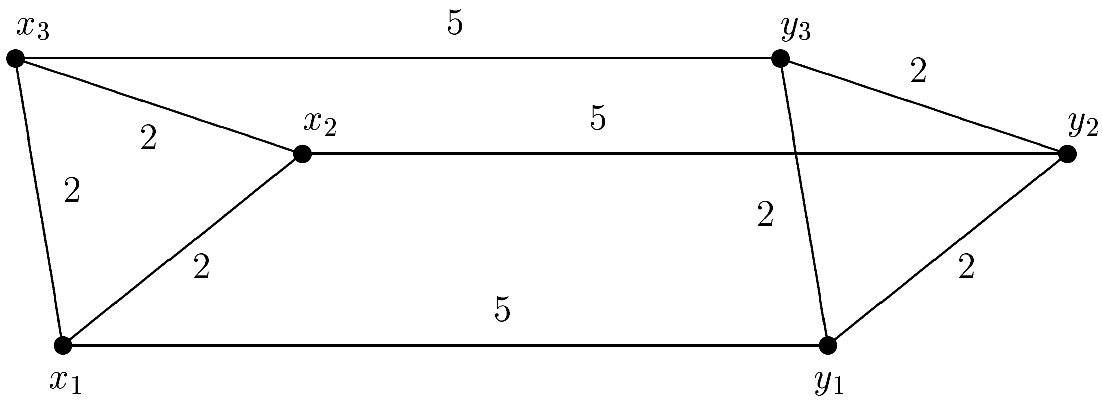

Figure 1. The space used in the proof of Theorem 1.