CS294-1 On-line Computation & Network Algorithms	**Spring 1997**

Lecture 6: February 6

Lecturer: Yair Bartal *Scribe: Anupam Gupta*

In this lecture, we look at further attempts to close the gap between the lower bounds for the k-server problem and known algorithms for this in an attempt to prove or disprove the k-server conjecture .

6.1 The k-server conjecture

Conjecture 6.1 *There is a k-competitive algorithm for the k-server problem in any metric space.*

Much progress has been made towards finding the truth of this conjecture:

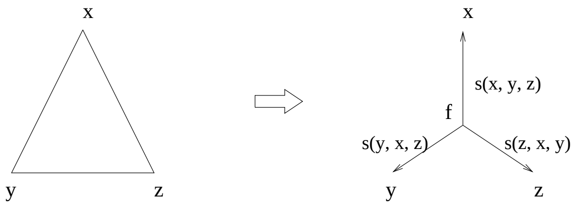
- Manasse, McGeogh and Sleator [MMS88], in their seminal paper gave the lower bound of k for all metric spaces.
- They also proved it for the special cases of metric spaces with $k + 1$ points, and for the case $k = 2$.
- It is known for the uniform metric (where it is equivalent to paging) [ST85].
- Chrobak et al.[CKPV90] proved it for the straight line, and later Chrobak and Larmore [CL91] generalized the result to trees.
- A major breakthrough came with Fiat, Rabani and Ravid [FRR90] who found an $O(k!^3)$-competitive algorithm, but for all metric spaces.
- This bound was improved via the Harmonic algorithm by Grove [Grove91] and later, by Bartal and Grove [BG92] to get a $O(2^k \log k)$-competitive randomized algorithm against adaptive adversaries.
- The next major breakthrough was by Koutsoupias and Papadimitriou [KP94], who show that the Work Function Algorithm is $2k - 1$-competitive.

6.1.1 The 2-Server Problem

There are several known algorithms that achieve an optimal competitive ratio of 2 for the 2-server problem. We will present here a simple approach due to Chrobak and Larmore [CL90].

There are a couple of ideas that enable us to give a 2-competitive algorithm that works for any metric space.

The first is that "*adding points to a metric space often makes it easier to work in and reason about them*". It must be noted that these imaginary points are not there in the metric space, and so the servers cannot move to those points, but must keep it in memory and perform further calculations as if they are really at that point.



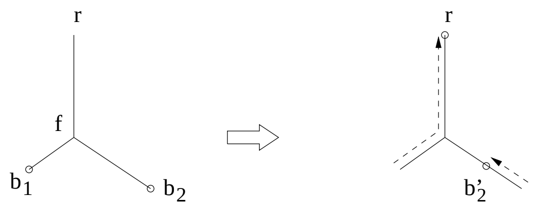
The second is the following construction. Suppose we have 3 points in the space x, y and z. If we perform the construction described in above figure, where we add to the space the point f, with $d(x, f) = s(x, y, z)$, $d(y, f) = s(y, x, z,)$ and $d(z, f) = s(z, x, y)$, where $s(x, y, z) = \frac{1}{2}(d(x, y) + d(x, z) - d(y, z))$ is called the slack of x, y and z. Note that distances between x, y and z remain the same as $d(x, y) = s(x, y, z) + s(y, x, z)$.

In the following, let b_1 and b_2 denote the position of the two servers, and let request be at point r. Construct the star as above, and add its points to the metric space (including points on its edges). Define the distance between every point on the star to x, y or z as the distance along the edges of the star, and to any other point $q \in \mathcal{M}$ by $\min_{i \in \{p,q,r\}}\{d(p, i) + d(i, q)\}$. It may be checked that the space with the points added is still a well defined metric space. Now we apply the 2-server algorithm for trees on the star.

Theorem 6.2 *The above algorithm is 2-competitive.*

Proof: We use the same potential function Φ as for the trees algorithm, and the analysis is very similar to the previous one.

- It is easy to see that the function is non-negative.
- Let the adversary move first and incur a cost x. Thus the cost of the minimum weight matching cannot exceed by more than x, and the Θ term remains unaffected. Thus $\Delta Phi \leq 2x$.
- Let the cost to the online algorithm be y. w.l.o.g., we can assume that b_1 is the one which serves the request. By the figure below, it is clear that b_1 moves a distance $d(b_1, r)$, while b_2 (which has now moved to b_2') has moved a distance $d(b_1, f) = s(b_1, b_2, r)$. Hence $y = d(b_1, r) + s(b_1, b_2, r)$.



The change in Θ: Let us look at the motion of the servers in two parts. The first is when both of them are moving, and then the distance between the two servers decreases by a distance $2s(b_1, b_2, r)$. The next part is when only b_1 moves, and then it increases by $s(r, b_1, b_2)$. Hence, $\Delta\Theta \leq -2s(b_1, b_2, r) + s(r, b_1, b_2)$.

The change in Ψ: Let M be the matching before the online algorithm moves its servers. Let the server used by the adversary to serve request r be a_1. Let M' be any matching and M_{opt} be the minimum weight matching after the request has been served. Let Ψ_W denote the cost of the matching W between the on-line and adversary servers. Then clearly, $\Psi_{M'} \geq \Psi_{M_{opt}}$, and hence $\Delta\Psi = \Psi_{M_{opt}} - \Psi_M \geq \Psi_{M'} - \Psi_M$. Thus we can bound the change in Ψ by choosing an appropriate matching M'.

CASE 1: $(a_1, b_1) \in M$. We let M' be the same matching as M, $\Psi_{M'} - \Psi_M \leq -d(b_1, r) + s(b_1, b_2, r) = -s(r, b_1, b_2)$.

CASE 2: $(a_2, b_1) \in M$. We now look at the movement of b_1 in two parts again. The first part is when both servers are moving. At this time, the distance between a_1 and b_2 decreases by $s(b_1, b_2, r)$, while that between b_1 and a_2 increases by at most that that same amount. Hence $\Delta\Psi \leq 0$. At this time, we can switch the matching to be $\{(a_1, b_1), (a_2, b_2)\}$ without affecting the weight. After this only b_1 is moving, and it reduces the weight of the matching by $s(r, b_1, b_2)$.

Hence the net $\Delta\Psi \leq -s(r, b_1, b_2)$. Hence, $\Delta\Phi = \Delta\Psi + \Delta\Theta \leq -2s(b_1, b_2, r) - s(r, b_1, b_2) = -s(b_1, b_2, r) - d(b_1, r) = -y$. ∎

Open Problem 6.3 *Find a randomized 2-server algorithm with a competitive ratio < 2 against an oblivious adversary.*

6.1.2 Optimal Off-line Algorithms

For any configuration A of the servers, let $w_t(A)$ be the optimal cost of serving the first t requests $\sigma_1\sigma_2\sigma_3\ldots\sigma_t$ and ending in the configuration A. Then,

$$w_{t+1}(A) = \min_X\{w_t(X) + d(X, A)\},$$

where the minimum is over all possible configurations X. The distance between two configurations is the weight of the minimum weight matching between them. This, in turn, gives

$$w_{t+1}(A) = \min_{a\in A}\{w_t(A + r - a) + d(a, r)\},$$

where r is the latest request made, namely $r = \sigma_{t+1}$, since any algorithm that serves all the requests must also serve r.

NOTE: An optimal off-line k-server algorithm can be computed in time polynomial in the length of the request sequence by reducing the problem to min-cost max-flow [CKPV90].

6.1.3 Metric Spaces with $k+1$ points

Let us consider the space to be the set of points $S = \{1, 2, \ldots, k+1\}$, with the distance function d. We shall return to the analysis of algorithms for $k+1$ points later in the course, but here is a brief overview.

The Balance algorithm works by trying to minimize the maximum distance moved my any server. we associate with each server i a cumulative mileage counter D_i. Upon a request at point r, we move server j such that $(D_j + d(s_j, r))$ is minimized, where s_j is the current position of the j^{th} server.

The Work Function algorithm uses the work function to guide its moves. Let B be the algorithm's current configuration. Then, we move the server at point s such that the value $w(B + r - s) + d(r, s)$ is minimized.

A careful consideration shows that the cost of the two algorithms differs by at most a constant additive term. Specifically, one can show by induction of the length of the request sequence that for all j there exists a constant $a)j$ (depending on the initial configuration) such that $D_j = w(S - s_j) - a_j$.

References

[BG92] Y. Bartal and E. Grove. The Harmonic k-Server Algorithm is Competitive, 1992. Accepted for publication in the JACM.

[CKPV90] M. Chrobak, H. Karloff, T. Payne, and S. Vishwanathan. Ne w Results on Server Problems. In *Proc. 1st ACM-SIAM Symp. on Discrete Algorithms*, pages 291-300, January 1990.

[CL90] M. Chrobak and L. Larmore. On Fast Algorithms for Two Servers. In *Proc. Mathematical Foundations of Computer Science*, Banská Bystrica, 1990. Also in *Journal of Algorithms.*

[CL91] M. Chrobak and L. Larmore. An Optimal On-Line k-Server Algorithm for Trees. *SIAM Journal of Computing*, 20:144-148, 1991.

[FRR90] A. Fiat, Y. Rabani, and Y. Ravid. Competitive k-Server Algorithms. In *Proc. of the 31st Ann. IEEE Symp. on Foundations of Computer Science*, pages 454-463, October 1990. Also in *JCSS Special FOCS Issue.*

[Grove91] E. Grove. The Harmonic k-Server Algorithm is Competitive. In *Proc. of the 23rd Ann. ACM Symp. on Theory of Computing*, pages 260-266, May 1991.

[KP94] E. Koutsoupias and C. Papadimitriou. On the k-Server Conjecture. In *Proc. of the 26th Ann. ACM Symp on Theory of Computing*, pages 507-511, May 1994.

[MMS88] M.S. Manasse, L.A. McGeoch, and D.D. Sleator. Competitive Algorithms for On-Line Problems. In *Proc. of the 20th Ann. ACM Symp. on Theory of Computing*, pages 322-333, May 1988.

[ST85] D.D. Sleator and R.E. Tarjan. Amortized Efficiency of List Update and Paging Rules. In *Communications of the ACM*, 28(2) pages 202-208, 1985.