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

Lecture 7: February 11

Lecturer: Yair Bartal *Scribe: Chris Umans*

In this lecture, we begin with a brief discussion of the applications of the k-server problem on trees, and then describe and analyze the harmonic k-server algorithm for general graphs.

7.1 Applications of the k-server problem on trees

7.1.1 Paging

Recall from the previous lectures that both paging and weighted paging can be represented as a k-server problem on stars. Of course a star is a tree, so we immediately have a k-competitive algorithm for paging and weighted paging.

7.1.2 Finite graphs/metric spaces

We can use the k-server algorithm on trees to derive an algorithm that operates on general finite graphs, but with a worse competitive ratio. Given a graph G, we construct its minimum spanning tree T and run the tree algorithm on T.

We note that $d_T(u, v) \leq (n-1)d(u, v)$, for all $u, v \in V(G)$, where $d_T(u, v)$ is the distance between u and v in T, and $d(u, v)$ is the distance in G. The adversary moves its servers in G; by forcing it to move in T, we increase its cost by $(n-1)$ at most, so $\mathrm{ADV}_T \leq (n-1)\mathrm{ADV}$. The competitive ratio of $k(n-1)$ for our algorithm (ALG) follows:

$$\mathrm{ALG} \leq \mathrm{ALG}_T \leq k\mathrm{ADV}_T \leq k(n-1)\mathrm{ADV}.$$

7.2 The harmonic k-server algorithm

In the harmonic k-server algorithm for a finite graph G, upon receipt of a request at r, we move each on-line server $x \in \mathrm{ON}$ with probability

$$\frac{1}{d(x, r)N(x)},$$

where $N(x)$ is a normalizing factor calculated as follows:

$$N(x) = \sum_{z \in \mathrm{ON}} \frac{1}{d(z, r)}.$$

For example, in the case of weighted paging, we move each server with probability inversely proportional to the weight of that page when the request appears in the center of the star.

A similar concept is that of a *harmonic random walk*. In a harmonic random walk, we move to a neighbor with probability inversely proportional to the distance. For the k-server problem with $k+1$ points, there is always a "hole" which is not covered by the algorithm's servers. Requests only matter is they come at that hole, so we can think of the harmonic algorithm as moving the hole in a harmonic random walk.

7.2.1 Hitting time

If the algorithm's hole is at point i and the adversary's hole is at point j, we define the *hitting time* H_{ij} to be the time required for our hole to coincide with the adversary's. It is known that

$$H_{ij} \leq \binom{k+1}{2} d(i,j),$$

and it is conjectured that $\binom{k+1}{2}$ is the best competitive ratio for the harmonic algorithm in general.

7.2.2 The harmonic k-server algorithm is competitive

We can define the potential function Φ to be the worst case cost of the harmonic algorithm against a lazy adversary. (A *lazy adversary* is one that hits the algorithm with requests at the point where the algorithm and the adversary do not coincide, until the coincide, at which point the lazy adversary moves a server so that they no longer coincide). If the adversary gives a request at point r (covered by the adversary), then $E(\Delta\Phi) = -E(\text{on-line cost})$. When the adversary moves one of its servers a distance x we want that $\Delta\Phi \leq cx$; this would imply that the algorithm is c-competitive. However this potential function is hard to analyze, so instead we use the matching method seen in previous lectures.

First, we construct a weighted bipartite graph on which the matching will be calculated. On one side, we have nodes for each point covered by the on-line servers; on the other, we have nodes for each point covered by the adversary. In general, the weight of the edge from x to y is assigned by choosing the minimum weight path from x to y that in its first step goes to some node on the adversary's side, and then moves among points on that side, as shown by the dotted line in figure 7.1. The successive segments of this path are assigned coefficients $c(1), c(2), \cdots, c(l)$:

$$w(x,y) = \min_{P(y)=\{y_1,y_2,\cdots,y_l=y\}} \left(c(1)d(x,y_1) + \sum_{1<i\leq l} c(i)d(y_{i-1},y_i) \right).$$

Alternatively, the weights can be defined as:

$$w(x,y) = \min_{P(y)=\{y_1,y_2,\cdots,y_l=y\}} \left(c(1)d(x,y_1) + \sum_{1<i\leq l} c(|\{y_1,\cdots,y_i\}|)d(y_{i-1},y_i) \right).$$

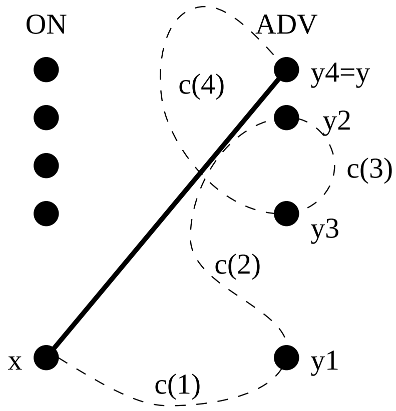

Figure 7.1: The matching used to define the potential function.

where $|\{y_1, \cdots, y_i\}|$ denotes any $y_1 \rightarrow y_i$ path. By the triangle inequality, these definitions are equivalent. Now, we define the potential function Φ as follows:

$$\Phi = \min_M \sum_{x \in \mathrm{ON}} w(x, M(x)),$$

where M ranges over all matchings. We now proceed to show the competitiveness of the harmonic algorithm. As noted above, we need to show 1) that $E(\Delta\Phi) \leq -E(\text{on-line cost})$ when the on-line algorithm moves a server, and 2) that $E(\Delta\Phi) \leq cE(\text{adversary cost})$ when the adversary moves.

Lemma 7.1 *When the on-line algorithm moves, $E(\Delta\Phi) \leq -E(ON)$.*

Proof: Assume that the request r being served comes at point $M(x)$. From the definition of the harmonic algorithm we have:

$$E(\mathrm{ON}) = \sum_{y \in \mathrm{ON}} \frac{1}{d(y, M(x))N(M(x))} d(y, M(x)) = \frac{k}{N(M(x))}.$$

The change in the potential function depends on the change in the weight of the matching for each $y \in \mathrm{ON}$, so we write:

$$E(\Delta\Phi) = \sum_{y \in \mathrm{ON}} \frac{1}{d(y, M(x))N(M(x))} \Delta\phi_y \leq \sum_{y \in \mathrm{ON}} \frac{A(y)}{N(M(x))}$$

where the last inequality holds because we will later show that $\Delta\phi_y \leq A(y)d(y, M(x))$ and finally that $\sum_{y \in \mathrm{ON}} A(y) = -k$.

We now calculate $\Delta\phi_y$; there are two cases to consider:

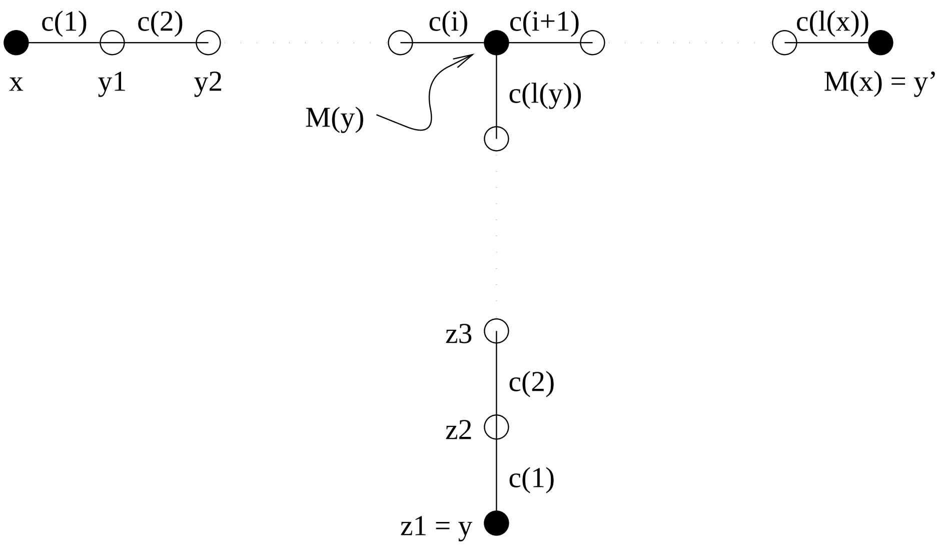

Figure 7.2: The configuration when $M(y) \in P(x)$.

- **Case 1:** $M(y) \in P(x)$, where $P(x)$ is the path between x and $M(x)$ with coefficients $c(1), c(2), \cdots, c(l(x))$. The configuration in this case is illustrated in figure 7.2. Here, the weight of the minimum matching decreases by at least the amount caused by matching y to $M(x)$ and x to $M(y)$, after moving server y to $y' = M(x)$. We first prove a small claim:

  Claim 7.2 *In figure 7.2, $l(y) < l(x)$, and in fact $l(y) \leq i$.*

  Proof: For the proof, we must first note that we will later choose the $c(i)$'s such that $c(1) \geq c(2) \geq c(3) \geq \cdots$. Now, we notice that if the claim were not true, we could change the matching and obtain a lesser weight:

$$\begin{aligned}&[w(y, M(x)) + w(x, M(y))] - [w(y, M(y)) + w(x, M(x))] \\ &\quad \leq \sum_{i<j\leq l(x)} (c(j + l(y) - i) - c(j))d(y_{j-1}, y_j) \leq 0.\end{aligned}$$

  The difference on the left is the difference between the proposed new matching and the current one, and if $l(y) > i$, it is ≤ 0, since $c(j + l(y) - i) \leq c(j)$. ■

  We now resume calculating $\Delta\phi_y$:

$$\Delta\phi_y = -\sum_{1\leq j<l(y)} c(j)d(z_j, z_{j+1}) - \sum_{i<jlel(x)} c(j)d(y_{j-1}, y_j).$$

  This is just a weighted sum of the segments on the path between y and $M(x)$. Since $l(y) < l(x)$ and the c's are decreasing, we can obtain the following bound:

$$\Delta\phi_y \leq -c(l(x))d(y, M(x)).$$

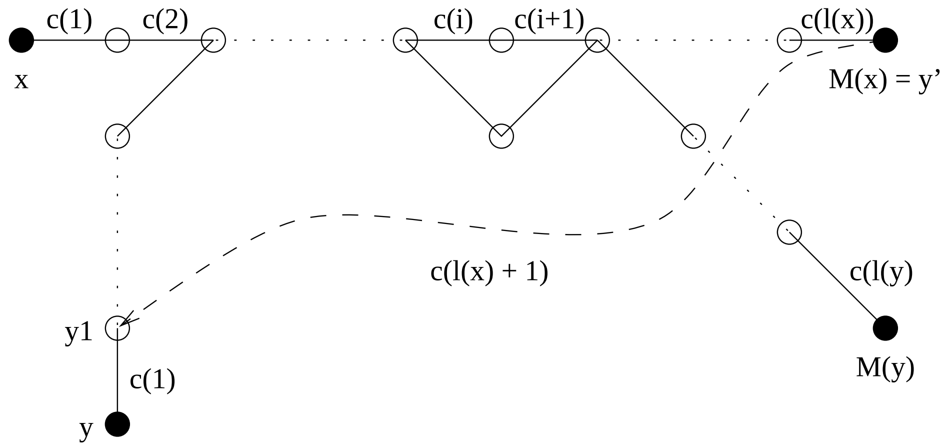

Figure 7.3: The configuration when $M(y) \notin P(x)$.

- **Case 2:** $M(y) \notin P(y)$. This is the configuration shown in figure 7.3. Here we are again interested in bounding the change in the weight of the matching caused by moving y to $y' = M(x)$ and matching y to $M(x)$ and x to $M(y)$.

  The new path that determines the weight of the edge matching x to $M(y)$ will follow $P(x)$ from x to $M(x)$, then go from $M(x)$ to y_1 (as shown by the dashed line in the figure), and finally follow $P(y)$ to $M(y)$.

  Consider $c(l(x)+j)$, the cost of the j_{th} edge in this path after $M(x)$. It is obvious that since the c's are decreasing, and $l(x)+j > j$, $c(l(x)+j) \leq c(j)$ for all j.

  The weight contributed by the edge $(x, M(y))$ in the new matching is greater than that contributed by the edge $(y, M(x))$ in the old matching, since additional segments from $P(y)$ now belong to the path from x to $M(y)$. However, the weight contributed by $P(y)$ is gone entirely, and as noted, the coefficients for each segment of $P(y)$ as used in the new path from x to $M(y)$ are less that the coefficients for the corresponding segments as used in $P(y)$ alone. So to get an upper bound, we simply assume that increase and decrease cancel, giving us:

  $$\Delta\phi_y \leq c(l(x)+1)d(M(x), y_1) - c(1)d(y, y_1) \leq c(l(x)+1)(d(M(x), y_1) - d(y, y_1)).$$

  By the triangle inequality, this simplifies to

  $$\Delta\phi_y \leq c(l(x)+1)d(y, M(x)).$$

As promised, we have shown that $\Delta\phi_y \leq A(y)d(y, M(x))$, where

$$A(y) = \begin{cases} -c(l(x)) & \text{if} M(y) \in P(y) \\ c(l(x)+1) & \text{if} M(y) \notin P(y). \end{cases}$$

The last step is to pick the c's so that $\sum_{y \in \mathrm{ON}} A(y) = -k$. This gives us the following constraint:

$$-l(x)c(l(x)) + (k - l(x))(c(l(x)+1)) = -k.$$

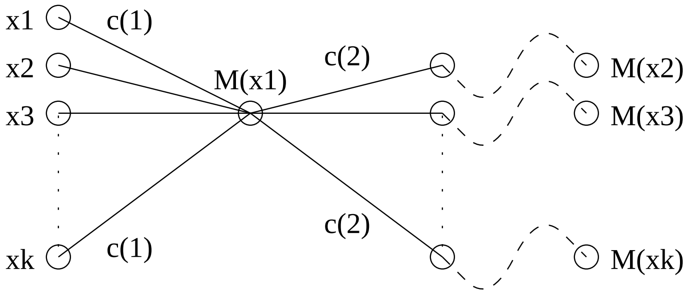

Figure 7.4: The worst case configuration for an adversary move.

We want this to hold for all possible path lengths, so we want that for all $1 \le j \le k$:

$$-j(c(j)) + (k-j)(c(j+1)) = -k.$$

Setting $c(k) = 1$ we get the following solution to this recurrence:

$$c(j) = (j-1)!(k-j)! \sum_{j \le m \le k} \frac{k}{m!(k-m)!},$$

so for example,

$$c(1) = (k-1)! \sum_{j \le m \le k} \frac{k}{m!(k-m)!} = 2^k - 1$$

$$c(2) = \frac{c(1)-k}{k-1} = \frac{2^k - 1 - k}{k-1}.$$

■

Lemma 7.3 *When the adversary moves, $E(\Delta\Phi) \le cE(\text{adversary cost})$.*

Proof: The worst case configuration is illustrated in figure 7.4. In this configuration, if the adversary moves $M(x_1)$ a distance d, it might increase the length of all of the solid edges in the figure by d, increasing the weight of the matching by $d(kc(1) + (k-1)c(2))$. Of course this is only an upper bound; a smalled weight matching might also be possible. If the factor $kc(1)+(k-1)c(2)$ is computed out using the values of $c(1)$ and $c(2)$ above, we get:

$$E(\Delta\Phi) \le ((k+1)(2^k-1)-k)E(\text{adversary cost}).$$

■

Theorem 7.4 *The harmonic algorithm is $((k+1)(2^k-1)-k)$-competitive.*

Proof: This follows from the previous two lemmas and the observation the $\Phi > 0$. ■

Finally we note that the ratio can be improved to $O(2^k \log k)$ by adjusting the analysis a bit.