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

Lecture 8: Feb 13

Lecturer: Yair Bartal *Scribe: Ashwin Nayak*

8.1 The work function algorithm for the k-server problem

In this lecture we study the work function algorithm for the k-server problem, and show that it is $2k-1$ competitive. The proof is due to Koutsoupias and Papadimitiou [KP94]. The potential function proof presented here is due to Yair Bartal.

8.1.1 Preliminaries

Recall that the k-server problem is defined as follows. k servers reside on some points in a metric space $\mathcal{M}$. At each time step, a request occurs at some point in $\mathcal{M}$, and is serviced by moving one of the servers to that point, if one is not present there already. The (multi) set of points on which the servers reside defines the *configuration* of the system. We denote configurations by the symbols $A, B, C, \ldots, A', B', C', \ldots$. Points in the metric space $\mathcal{M}$ are referred to by lower case letters $a, b, c \ldots$. The distance between two points $x, y \in \mathcal{M}$ is given by metric $d(x, y)$. Finally, $C + a$ and $C - a$ are used as abbreviations for $C \cup \{a\}$ and $C - \{a\}$ respectively.

8.1.2 The work function algorithm

Presumably, the simplest algorithm that one could come up for this problem is the *greedy* algorithm: for each point requested, if no server is currently residing at that point, move the server which is closest to it. Not surprisingly, the greedy algorithm is too naive to be competitive, as the example below shows.

Example: Consider the 2-server problem on a line. Let the two servers initially be at points b and c, as shown in Figure 8.1. The greedy algorithm moves a server back and forth between points a and b on the request sequence $a, b, a, b, a, b, a, b, \ldots$, while the optimum strategy would be to move the server at c to a. ■

The reason the greedy approach fails is because it fails to notice that when a particular region of the metric gets a lot of requests, it might be profitable to move more servers nearer to the region. One might try to remedy the forgetfulness of the greedy strategy by taking all the past request points into account while determining the next move. Looking at the sequence till now, we can determine the configuration that the optimum algorithm would be in after servicing the sequence. One might try to move close to this configuration, and at the same time try not to move very far from the current configuration, i.e. minimize the cost of moving to the next configuration while making it

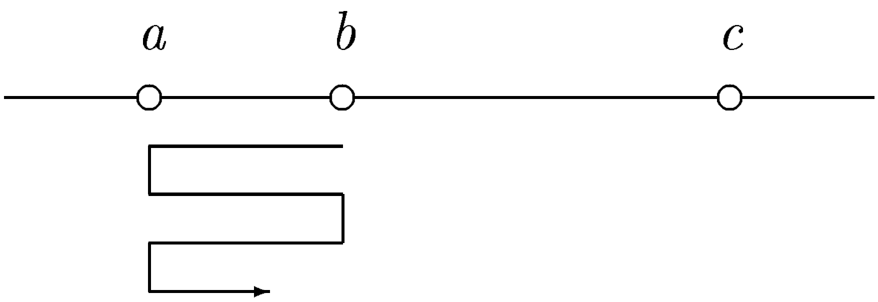

Figure 8.1: An example which shows that the greedy algorithm is not competitive.

as close to the optimal configuration as possible. This leads to the *work function* algorithm. We need a few definitions before we can describe the algorithm formally.

Definition 8.1 (Work function) *Consider an optimum algorithm starting in a configuration C_0 and servicing a sequence σ of requests and ending in a configuration C. The work function w_σ gives, for each configuration C, the cost incurred by the algorithm in servicing σ and ending in C.*

We will use the abbreviations w for w_σ, and w' for $w_{\sigma r}$ (the work function after servicing the request r), whenever σ is clear from the context.

We can now describe the work function algorithm.

Definition 8.2 (Work function algorithm) *Let σ be a request sequence, and let C be the configuration of the online algorithm after servicing σ. The work function algorithm serves the next request r by moving a server from that point c to r which minimizes $w'(C - c + r) + d(c, r)$.*

8.1.3 Basic properties of the work function

Consider the problem of computing the work function for a sequence of requests. Initially, $w(C)$ is simply the cost of moving from C_0 to C optimally. Note that this cost is the weight of the minimum weight matching in the complete bipartite graph between C_0 and C, in which edge weights are the shortest path distances between the points connected by the edges. We denote this "distance" between configurations C_1 and C_2 by $D(C_1, C_2)$. Suppose, inductively, that we have computed the work function w for serving the sequence σ, and would like to compute w', the work function after the next request r. For a configuration C, if $r \in C$, then $w'(C) = w(C)$. This is so because, clearly, $w'(C) \geq w(C)$ for any C. If $r \notin C$, then the server at r must've been moved to some point in c after the request was serviced. So in this case, $w'(C) = w'(C - c + r) + d(c, r) = w(C - c + r) + d(c, r)$ for some $c \in C$. In fact, the c must be such that $w(C - c + r) + d(c, r)$ is minimized. Note also that one way to serve requests in σ and end up in a configuration C is to first end up in configuration C', and then move the appropriate servers to C, so that $w(C) \leq w(C') + D(C, C')$. We summarise these facts below:

Fact 8.3 *For every configuration C, $w'(C) \geq w(C)$.*

Fact 8.4 *If $r \in C$, then $w'(C) = w(C)$.*

Fact 8.5 $w'(C) = \min_{c \in C}\{w(C - c + r) + d(c, r)\}$.

Note that this expression give the correct value for $w_{\sigma r}(C)$ even in the case $r \in C$.

Fact 8.6 $w(C) \leq w(C') + D(C, C')$.

8.1.4 Idea behind the analysis

Let us analyse the cost incurred by the work function algorithm in servicing requests from a sequence σ. Suppose the algorithm starts in the configuration C_0 and ends up in the configuration C_f. Each successive configuration of the algorithm is obtained by moving some server in the previous configuration to the current request point. We can assume that the final configuration C_f of the online algorithm is the same as that of the optimal offline algorithm. This is so because if it were not the case, we could extend the sequence σ with repeated requests from the final configuration of the optimal algorithm and force the online algorithm to move to that configuration. This would only increase the on-line cost, while leaving the oof-line cost unaffected.

Consider the online cost, $\mathrm{ON}(r)$, of servicing a request r after the sequence σ has been handled and the algorithm is in configuration C. The algorithm would have moved a server from some point c in C to r such that $w'(C') + d(c, r)$ is minimized, where $C' = C - c + r$. The cost of servicing r is thus $d(c, r)$. We need to compare the sum of the costs of such moves to the optimal off-line cost. To this end, we define a quantity called the *off-line pseudocost*, which is the difference $w'(C') - w(C)$. We denote this cost by $\mathrm{POFF}(r)$. The sum $\sum_r \mathrm{POFF}(r)$ over all requests of the off-line pseudocost is just the optimum off-line cost, since the value of the work function at the final configuration (which is the same as that of the off-line algorithm), is exactly this.

Now consider the sum of the on-line cost and the off-line pseudocost:

$$\mathrm{ON}(r) + \mathrm{POFF}(r) = w'(C') - w(C) + d(c, r).$$

From Fact 8.5, we get $w'(C') + d(c, r) = w'(C)$, since c minimizes $w'(C - c + r) + d(c, r)$. So,

$$\mathrm{ON}(r) + \mathrm{POFF}(r) = w'(C) - w(C)$$

which is bounded above by the maximum of this quantity over all configurations. I.e.

$$\mathrm{ON}(r) + \mathrm{POFF}(r) \leq \max_X\{w'(X) - w(X)\}.$$

Call the quantity on the right the *extended cost*, $\mathrm{EXT}(r)$, of servicing request r. The sum of the left hand side over all the requests is clearly the sum of the on-line cost (ON) and the optimal off-line cost (OPT), as argued above. Thus, if we are able to bound the total extended cost (EXT) in terms of the optimal off-line cost, we would be able to arrive at a competitive ratio for the work function algorithm. In particular, we have:

Fact 8.7 *If the total extended cost is bounded above by $(c+1)$ times the optimal off-line cost plus a constant, then the work function algorithm is c-competetive.*

8.1.5 Metric space over $k+1$ points

In the special case of a metric space over $k+1$ points, we can quickly show the work function algorithm is k-competitive through a potential function argument.

Consider the potential $\Phi = \sum_X w(X)$. When the request r is serviced, certainly, $\Delta\Phi \geq w'(A) - w(A)$, where A maximizes the difference $w'(X) - w(X)$. Thus, $\Delta\Phi \geq \text{EXT}(r)$, and so summing up over all the request points, $\Phi_f - \Phi_0 \geq \text{EXT}$. Now, $\Phi_f \leq (k+1)\max_X w_f(X)$, since there are exactly $(k+1)$ possible configurations. Let the maximum of $w_f(X)$ be achieved at A. Then, $w_f(A) \leq w_f(C_f) + D(A, C_f)$. So, $\Phi_f \leq (k+1)w_f(C_f) + (k+1)D(A, C_f)$. Note that the last term in this sum is bounded by $(k+1)$ times the maximum distance between any two points in the metric space, and hence is a constant. We have thus proved that

$$\text{EXT} \leq \Phi_f - \Phi_0 \leq (k+1)w_f(C_f) + \text{constant}$$

where the constant depends on the metric space. (Note that Φ_0 is also a constant.) Since $w_f(C_f)$ is the optimal cost, we have proved that the work function algorithm is k-competitive on a metric space with $k+1$ points, by Fact 8.7.

8.1.6 General metric space case

To derive a competitive ratio for general metric spaces, suppose we could define a function μ_σ on the metric space $\mathcal{M}$ such that the following properties hold: (we drop the subscript σ for convenience, as with the work function)

1. $\mu(x) \leq \text{OPT}$, for all $x \in C_f$.
2. $\mu_0(x) \geq -$constant for every $x \in C_0$.
3. $\mu'(x) \geq \mu(x)$, for all $x \in \mathcal{M}$.
4. $\mu(x) - \mu(y) \leq k \cdot d(x, y)$, for all $x, y \in \mathcal{M}$.
5. $\mu'(r) - \mu(r) = \text{EXT}(r)$, where r is the current request.

Given such a function μ, we could define a potential function Φ to get a bound on the total extended cost. Let U be the configuration of the optimal off-line algorithm at some stage of processing the requests. Then, Φ is defined to be:

$$\Phi = \sum_{u \in U} \mu(u).$$

We immediately get the following properties of Φ as a consequence of the properties of μ:

1. $\Phi_f \leq k \cdot \text{OPT}$, from Property 1, and the fact that there are exactly k servers in any configuration U.

2. $\Phi_0 \geq -$constant, from Property 2, and the fact that the off-line algorithm starts off in the configuration C_0.

3. When the off-line algorithm moves a server from u to v, then $\Delta\Phi \geq -k \cdot d(u, v)$. We can think of the optimum algorithm moving its servers *before* the next request comes in. Then, this follows immediately from Property 4.

4. When a request appears at r, $\Delta\Phi \geq \text{EXT}(r)$. This is so because when the request occurs, the adversary (the optimum algorithm) already has a server at r, and the potential function changes only because of the change in μ. This can then be inferred as a consequence of Properties 3 and 5.

Summing up all the changes in Φ due to moves of the adversary and due to the requests, we get the following inequalities:

$$\text{EXT} - k \cdot \text{OPT} \leq \Phi_f - \Phi_0 \leq k \cdot \text{OPT} + \text{constant}.$$

The first term in the lower bound comes from the requests, and the second from the moves of the adversary (which constitute the cost incurred by the adversary). The upper bound simply follows form the first two properties listed above. This proves that $\text{EXT} \leq 2k \cdot \text{OPT} + \text{constant}$, which gives us the competitive ratio of $2k - 1$ from Fact 8.7, modulo the construction of the function μ.

8.1.7 Construction of the function μ

We now unveil the function μ and prove that it satisfies the properties listed above.

Let $x \in \mathcal{M}$. Define μ on x as:

$$\mu(x) = \min_Y \{w(Y) - \sum_{y \in Y} d(x, y)\}.$$

Properties 1 and 2 of μ follows trivially. Consider Property 3. Fix $x \in \mathcal{M}$. Let Z be such that $\mu'(x) = w'(Z) - \sum_{z \in Z} d(x, z)$. We call such a Z a *minimizer* of x with respect to w'. Now, by the definition of μ,

$$\begin{aligned} \mu(x) &\leq w(Z) - \sum_{z \in Z} d(x, z) \\ &\leq w'(Z) - \sum_{z \in Z} d(x, z) \\ &= \mu'(x) \end{aligned}$$

The second inequality follows from Fact 8.3.

To prove Property 4, assume that Z is the minimizer of y with respect to w. Then, we get the following sequence of inequalities:

$$\begin{aligned}
\mu(x) - \mu(y) &\leq [w(Z) - \sum_{z \in Z} d(x,z)] - [w(Z) - \sum_{z \in Z} d(y,z)] \\
&= \sum_{z \in Z} [d(y,z) - d(x,z)] \\
&\leq k \cdot d(x,y)
\end{aligned}$$

The last line is a direct consequence of the triangle inquality and the fact that Z has k points.

The last property (no. 5) follows from the *Duality Lemma* we present and prove in the sequel.

8.1.8 Quasiconvexity of the work function

Before we prove the Duality Lemma, we derive a crucial property of the work function, called *quasiconvexity.*

Lemma 8.8 (Quasiconvexity Lemma) *For all configurations X, Y, and for all $x \in X$, there exists a $y \in Y$ such that*

$$w(X - x + y) + w(Y - y + x) \leq w(X) + w(Y).$$

Proof: The lemma is trivial is $x \in X \cap Y$ by letting y=x. If $x \in X - Y$ we will prove that there exists $y \in Y - X$ satisfying the lemma, by induction on the length of the sequence of requests. In the proof we use the notation A, B for the sets and a, b for the elements while X, Y, x, y will be kept for the induction hypothesis.

Initially, $w_0(A) = D(A, C_0)$ and $w_0(B) = D(B, C_0)$. Fix the two matchings at which these distances are attained. Every $c_j \in C_0$ is matched to some point $a_j \in A$ and some point $b_j \in B$. This defines a matching between points in A and B. Consider the bipartite graph defined by this matching, and start at the node corresponding to a following a path in the graph until we end in a point $b \in B - A$. It follows that this b satisfies the required property

$$w_0(A - a + b) + w_0(B - b + a) \leq w_0(A) + w_0(B).$$

Suppose the property holds for w. Let the next request point be r. We show that the property holds for w'. By Fact 8.5, we have that there exist $a' \in A$ and $b' \in B$ sucht that

$$w'(A) + w'(B) = w(A - a' + r) + w(B - b' + r) + d(a', r) + d(b', r). \tag{8.1}$$

Case (a). $a = a'$ and $b' \notin A$. Then let $b = b'$. We then have

$$\begin{aligned}
w(A - a' + r) + d(b', r) &\geq w'(A - a + b) \\
w(B - b' + r) + d(a', r) &\geq w'(B - b + a).
\end{aligned}$$

This together with eqaution (8.1) implies that the property holds in this case.

For the rest of the cases we use the induction hypothesis with $X = A - a' + r$, $Y = B - b' + r$ and some value $x \in A - a' - (B - b')$ so we get that there exists $y \in B - b' - (A - a') \subseteq B - A$ (since $a' \notin B$) such that

$$w(A - a' + r) + w(B - b' + r) \geq w(A - a' + r - x + y) + w(B - b' + r - y + x). \tag{8.2}$$

Case (b). $a = a'$ and $b' \in A$. We use $x = b'$ and let $b = y$. We get that

$$\begin{aligned} w(A - a' + r - x + y) + d(b', r) &= w'(A - a + r - b' + b) + d(b', r) \geq w'(A - a + b) \\ w(B - b' + r - y + x) + d(a', r) &= w'(B - b' + r - b + b') + d(a, r) \geq w'(B - b + a). \end{aligned}$$

Case (c). $a \neq a'$. We use $x = a$ and let $b = y$. We get that

$$\begin{aligned} w(A - a' + r - x + y) + d(a', r) &= w'(A - a' + r - a + b) + d(a', r) \geq w'(A - a + b) \\ w(B - b' + r - y + x) + d(b', r) &= w'(B - b' + r - b + a) + d(b', r) \geq w'(B - b + a). \end{aligned}$$

The claim follows from inequalities (8.1) and (8.2) and the above case analysis:

$$\begin{aligned} w'(A) + w'(B) &\geq w'(A - a' + r - x + y) + w'(B - b' + r - y + x) + d(a', r) + d(b', r) \\ &\geq w'(A - a + b) + w'(B - b + a). \end{aligned}$$

■

8.1.9 The duality lemma

In this section, we present and prove two lemmata which constitute the Duality Lemma, and we use these to prove Property 5 of the function μ.

Lemma 8.9 *If A is a minimizer of r with respect to w, then A is a minimizer of r with respect to w'.*

Proof: Recall that A is said to be the minimizer of r with respect to w if $\mu(r) = w(A) - \sum_{a \in A} d(a, r)$. So it is enough to show that for all configurations B,

$$w'(B) - \sum_{b \in B} d(b, r) \geq w'(A) - \sum_{a \in A} d(a, r).$$

Equivalently, we could show that:

$$w'(B) - \sum_{b \in B} d(b, r) + w(A) \geq w'(A) - \sum_{a \in A} d(a, r) + w(A).$$

From Fact 8.5, we get that there exists $b' \in B$ such that

$$w'(B) = w(B - b' + r) + d(b', r).$$

Using quasiconvexity, we get that there exists $a' \in A$ such that

$$w(B - b' + r) + w(A) \quad \leq \quad w(B - b' + a') + w(A - a' + r).$$

Also, since A is a minimizer of r,

$$w(B - b' + a') - \sum_{b \in B - b' + a'} d(b, r) \quad \geq \quad w(A) - \sum_{a \in A} d(a, r).$$

Combining all these inequalities, we get:

$$\begin{aligned}
& w'(B) + w(A) - \sum_{b \in B} d(b, r) \\
= \quad & w(B - b' + r) + d(r, b') + w(A) - \sum_{b \in B} d(b, r) \\
= \quad & w(B - b' + r) + w(A) - \sum_{b \in B - b' + r} d(b, r) \\
\geq \quad & w(B - b' + a') + w(A - a' + r) - \sum_{b \in B - b' + r} d(b, r) \\
= \quad & w(B - b' + a') + w(A - a' + r) + d(r, a') - \sum_{b \in B - b' + a} d(b, r) \\
\geq \quad & w(A) + w(A - a' + r) + d(r, a') - \sum_{a \in A} d(a, r) \\
\geq \quad & w(A) + w'(A) - \sum_{a \in A} d(a, r)
\end{aligned}$$

where the last inequality is based on Fact 8.5. This completes the proof of the lemma. ∎

The second part of the Duality Lemma is as below:

Lemma 8.10 *If A is the minimizer of r with respect to w, then the extended cost occurs at A, i.e.,*

$$w'(A) - w(A) \quad = \quad \max_X \{w'(X) - w(X)\}$$

Proof: It suffices to show that for all configurations B,

$$w'(A) + w(B) \quad \geq \quad w'(B) + w(A).$$

Fact 8.5 gives us that there is $a' \in A$ such that

$$w'(A) \quad = \quad w(A - a' + r) + d(a', r).$$

Furthermore, invoking quasiconvexity, we get that there is a $b' \in B$ such that

$$w(A - a' + r) + w(B) \quad \geq \quad w(A - a' + b') + w(B - b' + r).$$

Finally, since A is a minimizer of r,

$$w(A - a' + b') - \sum_{a \in A - a' + b'} d(a, r) \quad \geq \quad w(A) - \sum_{a \in A} d(a, r)$$

which reduces to

$$w(A - a' + b') + d(a', r) \geq w(A) + d(b', r).$$

Putting all these together, we get:

$$\begin{aligned} w'(A) + w(B) &= w(A - a' + r) + d(a', r) + w(B) \\ &\geq w(A - a' + b') + d(a', r) + w(B - b' + r) \\ &\geq w(A) + d(b', r) + w(B - b' + r) \\ &\geq w(A) + w'(B). \end{aligned}$$

Again, the last inequality is based on Fact 8.5. ■

It is easy to see that Property 5 follows from these lemmata by using the definition of $\mu(r)$ and of $\text{EXT}(r)$. This completes the proof of the $2k - 1$ competitiveness of the work function algorithm.

8.1.10 Open problems

The result presented above still leaves a lot of questions unanswered. We list them below for further work.

1. Can a competitive ratio of k be achieved for arbitrary metric spaces?
2. It is known that the work function algorithm is k-competitive for metric spaces with $k + 2$ points, the line, and for the weighted cache problem. Can these results be extended to other special classes of metric spaces such as trees or cycles?
3. Can it be proved that the work function algorithm is not k-competitive for some classes of metric spaces?
4. Implementing the work function algorithm is highly inefficient. Are there simpler algorithms that achieve poly(k)-competitiveness, maybe for special cases such as cycles, or in the plane?
5. Can the proof for the 2-competitiveness of the work function algoeithm for the 2-server case be simplified?
6. The Harmonic algorithm is known to be $O(2^k \log k)$-competitive. Can it be shown to be $O(k^2)$-competitive?

References

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