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

Lecture 14: March 13

Lecturer: Yair Bartal *Scribe: Zhendong Su*

In this lecture we talk about the k-page migration problem and the Metrical Task Systems and Relaxed Metrical Task Systems.

14.1 K-page Migration Problem

The k-page Migration Problem [BCI97] is a generalization of the Page Migration Problem we talked about before. In the k-page migration problem, there are k copies of a file in a network instead of just one copy in the standard page migration problem. A request from one of the nodes in the network costs the distance to the nearest copy of the file. At some point, the copies of the file may also be migrated to a node at the cost of D times of the distance between the two nodes. The problem is to minimize the total access cost and migration cost.

We know of good algorithm for solving the Page Migration Problem. Can we generalize them for the k-page Migration Problem? There are a few straight forward ways of doing this. However, none of them is good.

Instead, we approach this problem differently. First we observe some similarity between the k-page Migration Problem and the k-erver problem. In addition, we know some good algorithms for the k-server problem. We may somehow reduce the k-page Migration problem to the k-server problem. So, the idea is to use good algorithms for the k-server problem to get a good algorithm for the k-page migration problem.

Recall in the k-server problem, we have a metric space and k servers located in some k points of the metric space. There is a sequence of requests $\sigma = \sigma_1\sigma_2\ldots$, where σ_i is a point in the metric space. For each request σ_i, the online algorithm should serve the request by moving one of the k servers at a cost of the distance travelled. The problem is to minimize the total service cost.

We can formulate the problem in the following way. Let a configuration of be the positions of the k servers and define $dist(C, C')$ as the cost of changing configuration from C to C', where C and C' are two configurations. To serve a request r in configuration C, the online algorithm:

- move a server to the request,
- if C has a server at r, then $cost(C, r) = 0$, otherwise, $cost(C, r) = \infty$.

The k-page migration can also be formulated similarly. Let a configuration be the positions of the k pages and define

$$dist_D(C, C') = D \cdot dist(C, C').$$

To serve a request r in configuration C, we define the request service cost

$$cost_D(C, r) = \min_{C'}(dist(C, C') + cost(C', r)).$$

This recasted the k-page migration problem into the k-server problem.

14.2 Metrical Task Systems and Relaxed Metrical Task Systems

In this section, we talk about a general framework for a large class of on-line problems including the k-server and k-page migration problems.

14.2.1 Defintion of Metrical Task Systems

A Metrical Task System consists of:

- C: a set of configurations
- $dist(C_1, C_2)$: configuration change cost from C_1 to C_2
- $(C, dist)$ form a metric space
- $task(C, r)$: cost of servicing r in C.
- Objective: minimize the sum of the configuation change cost and the request service cost

Given a specific task system, we can define a corresponding relaxed system:

14.2.2 Definition of D-relaxed Task Systems

A D-relaxed Task System consists of:

- C: a set of configurations
- $dist_D(C_1, C_2) = Ddist(C_1, C_2)$, where $dist$ is defined as in Metrical Task Systems
- $(C, dist_D)$ clearly also form a metric space
- $task_D(C, r) = min_{C'}\{dist(C, C') + task(C', r)\}$

¿From the above formulations of the k-server problem and the k-page migration problem, we notice that they are instances of MTS (Metrical Task Systems) and DRTS (D-relaxed Task Systems) respectively. The Steiner-tree problem is an instance of the MTS. The page-replication problem is a DRTS with respect to the Steiner-tree problem.

Suppose we have a c-competitve algorithm for a MTS. We can easily construct a $O(cD)$-competitive algorithm for a DRTS. However, this is not good enough, since D might be large.

We have the following results for the general relaxed task system model:

Theorem 14.1 *Given a c-competitive deterministic algorithm for a MTS, we can construct a $9c^2$-competitive algorithm for its corresponding DRTS.*

Proof: Let A be a c-competitive deterministic algorithm for a MTS. Consider the following algorithm:

1. The algorithm simulates $2D$ copies of A. Let them be $A_1, A_2, \ldots, A_{2D}$. During each request, only of the A_i serves the request.
2. The algorithm maintains the same configuration as A_1.
3. A request r is given to
   - $A_i, i \geq 2$, if $cost_{A_i} \geq \frac{1}{3c} cost_{A_1}(r)$; (in this case, the configuration of A_i is updated, but the algorithm serves r remotely. Its configuration is not changed.)
   - else the request is given to A_1. (in this case, the algorithm serves r and moves to the new configuration of A_1.)

We can show that this algorithm is $9c^2$-competitive.

The proof consists of two parts:

Lemma 14.2 *Let σ be a sequence of requests, and let $\sigma_1 \sigma_2 \ldots \sigma_{2D}$ be subsequences of σ that are serviced by $A_1, A_2, \ldots, A_{2D}$ respectively. Also, we let $cost^D_{OPT}(\sigma)$ be the optimal off-line cost of servicing σ in the DRTS. Then*

$$\sum_{i=1}^{2D} cost_{A_i}(\sigma_i) \leq 2c \cdot cost^D_{OPT}(\sigma).$$

Proof: Since the A_i's are c-competitive, we have

$$\sum_{i=1}^{2D} cost_{A_i}(\sigma_i) \leq c \cdot \sum_{i=1}^{2D} cost_{opt_i}(\sigma_i),$$

where $cost_{opt_i}(\sigma_i)$ is the optimal offline cost of servicing σ_i in the MTS. Thus, it is sufficient to show that

$$\sum_{i=1}^{2D} cost_{opt_i}(\sigma_i) \leq 2 \cdot cost^D_{OPT}(\sigma).$$

We define opt'_i to service each σ_i in the DRTS. All the opt'_i maintain the same configuration as OPT.

- When OPT changes configuration from C_1 to C_2, all opt_i also changes configuration from C_1 to C_2. Each opt_i has a cost of $dist(C_1, C_2)$. Also $cost^D_{OPT} = dist_D(C_1, C_2) = D \cdot dist(C_1, C_2)$. Thus, we have

$$\begin{aligned}\sum_{i=1}^{2D} cost_{opt'_i}(\sigma_i) &= 2D \cdot dist(C_1, C_2) \\ &= 2(D \cdot dist(C_1, C_2) \\ &= 2 \cdot cost^D_{OPT}(\sigma).\end{aligned}$$

- When OPT serves a request r from C, let opt'_i be the one that serves r, i.e., r is in σ_i. Suppose opt'_i moves from C to C' to serve r and then moves back to C. Thus, we have, $cost_{opt'_i} = 2 \cdot dist(C, C') + task(C', r)$. The cost of OPT is $task_D(C, r) = dist(C, C') + task(C', r)$. Hence, we have

$$cost_{opt'_i} \leq 2 \cdot cost^D_{OPT}.$$

Therefore, we have

$$\sum_{i=1}^{2D} cost_{opt'_i}(\sigma_i) \leq 2 \cdot cost^D_{OPT}(\sigma).$$

Also, since opt_i's are optimal offline costs, we have

$$\sum_{i=1}^{2D} cost_{opt_i}(\sigma_i) \leq \sum_{i=1}^{2D} cost_{opt'_i}(\sigma_i).$$

Thus, the lemma follows. ∎

Lemma 14.3 *Let σ_i be the sequence of requests given to A_i while running the algorithm. Then we have*

$$cost_{Alg}(\sigma) \leq 4.5c \sum_{i=1}^{2D} cost_{A_i}(\sigma_i).$$

Proof: We split the cost of the algorithm into $cost^S_{Alg}(\sigma)$, the cost of the algorithm on the requests that are given to A_1 and $cost^M_{Alg}(\sigma)$,the cost on the requests that are given to A_i with $i \geq 2$. Consider the cost the algorithm on a request of r. We have two cases.

- If r is given to $A_i, i \geq 2$, we know that $cost_{A_i}(r) \geq \frac{1}{3c} cost_{A_1}(r)$, which implies that

$$\begin{aligned}cost_{Alg}(r) &\leq cost_{A_1}(r) \\ &\leq cost_{A_i}(r).\end{aligned}$$

  Thus, we can bound $cost^S_{Alg}(\sigma)$.

$$cost^S_{Alg}(\sigma) = \sum_{r \text{ given to } A_i, i \geq 2} cost_{Alg}(r)$$

$$
\begin{aligned}
&\le 3c \sum_r \sum_{i=2}^{2D} cost_{A_i}(r) \\
&\le 3c \sum_r \sum_{i=1}^{2D} cost_{A_i}(r) \\
&= 3c \sum_{i=1}^{2D} cost_{A_i}(\sigma_i).
\end{aligned}
$$

- If r is given to A_1, we first notice that $cost_{Alg}(r) \le D \cdot cost_{A_1}(r)$. This follows from $cost_{A_1}(r) = dist(C, C') + task(C', r)$ and $cost_{Alg}(r) = D \cdot dist(C, C') + task(C', r)$. Thus, to prove the lemma, it is sufficient to bound $cost_{A_1}(\sigma_1)$ in terms of $\sum_{i=1}^{2D} cost_{A_i}(\sigma_i)$. Consider algorithm A'_i that simulates A_i on σ_i, and also serve all requests from σ_1 as the following:

  If a request r is from σ_i, A'_i moves from its current configuration C to C', serves r, and moves back to C.

  We have

$$
\begin{aligned}
cost_{A'_i}(r) &= 2 \cdot dist(C, C') + task(C', r) \\
&\le 2 \cdot cost_{A_i}(r).
\end{aligned}
$$

  Since r is given to A_1, $cost_{A_i}(r) \le \frac{1}{3c} cost_{A_1}(r)$ and $cost_{A'_i}(r) \le \frac{2}{3c} cost_{A_1}(r)$. Thus,

$$
\begin{aligned}
cost_{A'_i} &\le cost_{A_i}(\sigma_i) + \frac{2}{3c} \sum_{r \text{ given to } A_1} cost_{A_1}(r) \\
&= cost_{A_i}(\sigma_i) + \frac{2}{3c} cost_{A_1}(\sigma_1)
\end{aligned}
$$

  We know that $cost_{A_1}(\sigma_1) \le c \cdot cost_{A'_i}(\sigma_1)$ since A_1 is c-competitive. Hence,

$$
\frac{1}{c} cost_{A_1}(\sigma_1) \le cost_{A'_i}(\sigma_1) \le cost_{A_i}(\sigma_1) + \frac{2}{3c} cost_{A_1}(\sigma_1).
$$

  Therefore,

$$
cost_{A_1}(\sigma_1) \le 3c \cdot cost_{A_i}(\sigma_i).
$$

  Thus,

$$
\begin{aligned}
cost^M_{Alg}(\sigma) &\le D \cdot cost_{A_1}(\sigma_1) \\
&= \frac{1}{2} \cdot 2D cost_{A_1}(\sigma_1) \\
&= \frac{1}{2} \sum_{i=1}^{2D} cost_{A_1}(\sigma_1) \\
&\le \frac{3c}{2} \sum_{i=1}^{2D} cost_{A_i}(\sigma_i) \\
&= 1.5c \sum_{i=1}^{2D} cost_{A_i}(\sigma_i)
\end{aligned}
$$

Therefore,

$$cost_{Alg}(\sigma) \leq 4.5c \sum_{i=1}^{2D} cost_{A_i}(\sigma_i).$$

■

The theorem follows immediately from the two lemmas. ■

Theorem 14.4 *Given a c-competitve randomized algorithm for a MTS against oblivious adversaries, we can construct a randomized 3c-competitive algorithm for a DRTS against oblivious adversaries.*

Proof: Let A be a c-competitve randomized algorithm for the MTS against oblivious adversaries. We can define a randomized version of the above algorithm and show that it is $3c$-competitive against an oblivious adversary. The algorithm also simulates $2D$ copies of algorithm A. Let them be $A_1, A_2, \ldots, A_{2D}$. At the beginning the algorithm chooses one of the $2D$ copies uniformly at random, say A_i and then always maintains the same configuration as A_i. The requests are always given to the copy of the alogrithm with the highest cost of serving the requests. This algorithm is $3c$-competitive against an oblivious adversary. The analysis is similar to that of the derterministic case. ■

References

[BCI97] Y. Bartal, M. Charikar and P. Indyk. On Page Migration and Other Relaxed Task Systems. In *Proc. of the 8th Ann. ACM-SIAM Symp. on Discrete Algorithms*, January 1997.