CS880: Approximation and Online Algorithms	**Scribe:** Xiating Ouyang
Lecture 21: Metric Task System	**Date:** 11/11/2019

In this lecture, we show that no algorithm can obtain a competitive ratio of less than $2N-1$ for the MetricTaskSystem problem. We then present the work function algorithm, which achieves a competitive ratio of $2N-1$ for the problem. In the end we introduce the Experts problem.

21.1 Metric Task System

The MetricTaskSystem problem (MTS) is defined as follows: Given a metric space over N points and a request sequence $\sigma = (c^1, c^2, \ldots, c^T)$ of cost vectors $c^t \in \mathbb{R}^N$, determine a sequence of locations $(i_1, i_2, \ldots, i_T)$ with $i_t \in [N]$ to minimize

$$\sum_{t \in [T]} d(i_{t-1}, i_t) + c^t_{i_t}.$$

Let w_i^t be the minimum cost of serving the tasks up until step t and finishes at location i. Then we have

$$w_i^t = \min_{i \in [N]} \min_{(i_1, i_2, \ldots, i_t) \in [T]} \{\sum_{t' \leq t} (d(i_{t'-1}, i_{t'}) + c^{t'}_{i_{t'}}) + d(i_t, i)\},$$

and that

$$OPT = \min_{i \in [N]} w_i^T.$$

Consider the work function algorithm: At each step t, move to

$$i_t = \arg\min_{i \in [N]} w_i^t + d(i_{t-1}, i).$$

In the following sections we first show that MTC does not admit an algorithm achieving a competitive ratio less than $2N-1$, and then show that the work function algorithm achieves a competitive ratio of $2N-1$.

21.1.1 Hardness

We show the following theorem.

Theorem 21.1.1 *No algorithm can achieve a competitive ratio of less than $2N-1$ for MTC on any metric space of size N.*

To show , we first argue that it suffices to consider cost vectors only assigning ε to one component and 0 elsewhere.

Lemma 21.1.2 *It suffices to assume that every cost vector to the MTC problem is of the form $(0, 0, \ldots, \varepsilon, 0, \ldots, 0)$, where $c_i^t = \varepsilon$ for just one $i = i_t^*$.*

Proof: This is a proof sketch. We may pick an appropriate ε, and then for any i and t, we decompose each cost vector c_i^t into

$$c_i^t = \frac{c_1^t}{\varepsilon}(\varepsilon, 0, \dots, 0) + \frac{c_2^t}{\varepsilon}(0, \varepsilon, 0, \dots, 0) + \dots.$$

Then the offline optimal algorithm achieves no worse than before, since the original solution still works; the online algorithm however loses information than before, and thus performs no better. Therefore, the competitive ratio under this special class of cost vectors is no worse than the original competitive ratio the algorithm would obtain. ■

We now present the proof of Theorem 21.1.1.

Proof of Theorem 21.1.1: Consider the cruel adversary that sets $c_i^t = \begin{cases} \varepsilon & \text{if } i = i_{t-1} \\ 0 & \text{if } i \neq i_{t-1}. \end{cases}$ We assume without loss of generality that the metric space satisfies that $d(i, j) \in [1, D]$ for $i \neq j$.

We now place $2N - 1$ servers in the metric space, and for any step t, we keep only one server at i_t and two servers at each of the other $N - 1$ locations—with a total of $2N - 1$ servers. Let k be the number of step t with $i_t \neq t_{t+1}$.

Thus the cost of the online algorithm is

$$ALG = \sum_t d(i_{t+1}, i_t) + (T - k)\varepsilon.$$

The total cost of the $2N - 1$ servers is

$$SUM = \sum_t d(i_{t+1}, i_t) + T\varepsilon \leq ALG + k\varepsilon \leq ALG + ALG\epsilon,$$

where the first inequality holds by construction, and the second inequality holds since at each step, the algorithm incurs at least a cost of 1 from the metric space, and thus the cost of the algorithm is at least k. Therefore there exists a server (among the $2N - 1$ servers) that has a cost less than

$$\frac{1 + \varepsilon}{2N - 1} ALG,$$

and taking $\varepsilon \to 0$ would complete the proof. ■

21.1.2 Work function algorithm: $2N - 1$ is tight

The following properties hold for work function w_i^t by construction:

1. w_i^t is nondecreasing;
2. $w_i^t = \min_{i'}\{w_{i'}^{t-1} + c_{i'}^t + d(i', i)\}$, which implies that $w_i^t \leq w_{i'}^t + d(i', i)$ for all $i \neq i'$; and
3. $w_i^t = \min_i(w_{i'}^t + d(i', i))$.

We analyze the performance of the work function algorithm by a potential function $\Phi^t = w^t_{i_t} + 2\sum_{i\neq i_t} w^t_i$.

Claim 21.1.3 *The cost of the algorithm at step t is at most $\Phi^t - \Phi^{t-1}$.*

Claim 21.1.4 $\Phi^T \leq (2N-1)(OPT + D)$.

Combining Claim 21.1.3 and 21.1.4, it thus follows that

$$ALG \leq \sum_t (\Phi^t - \Phi^{t-1}) \leq \Phi^T \leq (2N-1)(OPT + D),$$

implying that that the work function algorithm achieves a competitive ratio of $2N-1$ (ignoring the additive constant D term).

It hence suffices to prove Claim 21.1.3 and 21.1.4.

Proof of Claim 21.1.4: By construction, $\Phi^T = w^T_{i_T} + 2\sum_{i\neq i_T} w^T_i$, and by property 2 we have $w^T_i \leq w^T_{i^*_T} + d(i^*_T, i) \leq OPT + D$ for the optimal ending location i^*_T. Hence

$$\Phi^T \leq (2N-1)(OPT + D).$$

■

Proof of Claim 21.1.3: Consider the algorithm is at step $t-1$ and the adversary places the cost vector c^t with ε at i_{t-1} and 0 elsewhere. There are two cases:

Case 1 : Suppose that $i_{t-1} = i_t$, that is, the algorithm does not move its location and thus incurs a cost of ε. We thus have that

$$\begin{aligned} w^t_{i_t} &= \min_{i'}\{w^{t-1}_{i'} + c^t_{i'} + d(i_{t-1}, i')\} \\ &= \min\{w^{t-1}_{i_t}, \min_{j\neq i_t}\{w^t_j + d(i_{t-1}, j)\}\}. \end{aligned}$$

Since $i_{t-1} = i_t$, we have that $w^{t-1}_{i_t} \leq \min_{j\neq i_t}\{w^t_j + d(i_{t-1}, j)\}$, or else the algorithm would have moved to some j with $j \neq i_{t-1}$, a contradiction. Thus $w^t_{i_t} = w^{t-1}_{i_t} + \varepsilon$, that is

$$\Phi^t - \Phi^{t-1} \geq \varepsilon = ALG.$$

Case 2 : Suppose that the algorithm moves to some location $i_t = j$ with $i_{t-1} \neq i_t$. In this case, the cost of algorithm incurred is $d(i_{t-1}, j)$. Thus in this case, we have that

$$\begin{aligned} \Phi^t - \Phi^{t-1} &\geq w^t_j + 2w^t_{i_{t-1}} - 2w^{t-1}_j - w^{t-1}_{i_{t-1}} \\ &\geq w^t_{i_{t-1}} - w^{t-1}_j \\ &\geq w^t_j + d(i_{t-1}, j) - w^{t-1}_j \\ &\geq d(i_{t-1}, j) = ALG, \end{aligned}$$

where the first inequality holds by expanding the definition of Φ; the second inequality holds since $w^t_j \geq w^{t-1}_j$ and $w^t_{i_{t-1}} \geq w^{t-1}_{i_{t-1}}$; the third inequality holds since $j \neq i_{t-1}$ and thus $w^t_{i_{t-1}} \geq w^t_j + d(i_{t-1}, j)$; and the forth inequality holds since $w^t_j \geq w^{t-1}_j$.

Hence the cost of the algorithm at each step t is at most $\Phi^t - \Phi^{t-1}$. ■

It is worth mentioning that the work function algorithm can also be applied to the K-SERVER problem, and yield a competitive ratio of $2k-1$. In the K-SERVER, the cost vector contains either 0 or ∞ for each component.

21.2 Expert Problem

The expert problem is defined as follows: Given N experts and a sequence of cost vectors $(c^1, c^2, \ldots, c^T)$ with $c^T \in [0,1]^N$. At every step, online algorithm chooses $i_t \in [N]$ **before** observing c^t, and the total cost of the algorithm is defined as

$$ALG = \sum_t c^t_{i_t}.$$

The cost of the optimal expert is

$$OPT = \min_i \sum_t c^t_i,$$

and the regret of the algorithm is

$$ALG - OPT = \sum_t c^t_{i_t} - \sum_t c^t_{i^*},$$

where $i^* = \arg\min \sum_t c^t_i$. The objective is to minimize the regret.

We consider exponential weighted update algorithm: At each step t, we assign a probability p^t_i to each expert i as the probability of picking expert i at step t. We update the probability using $p^{t+1}_i = p^t_i e^{-\eta c^t_i}$ for some learning rate η and then normalize each p^{t+1}_i to obtain the distribution of $\tilde{p}^{t+1}_i$ at each step. The cost of the algorithm is thus revised as

$$\mathbb{E}[ALG] = \sum_{t,i} \tilde{p}^t_i c^t_{i_t}.$$