ORIE 6334 Spectral Graph Theory October 20, 2016

Lecture 17

Lecturer: David P. Williamson *Scribe: Yingjie Bi*

1 Approximate Potentials from Approximate Flow

In the last lecture, we presented a combinatorial algorithm to find potentials p that solve the Laplacian system $L_G p = b$. The electrical flow f_* minimizes the energy

$$\mathcal{E}(f) = \sum_{\{i,j\} \in E} f^2(i,j) r(i,j).$$

We showed that in each iteration the expected decrease in energy for the maintained flow f_k is

$$\mathbf{E}[\mathcal{E}(f_k) - \mathcal{E}(f_{k+1})] = \frac{1}{\tau}\mathbf{E}[\text{gap}(f_k, p_k)] \geq \frac{1}{\tau}\mathbf{E}[\mathcal{E}(f_k) - \mathcal{E}(f_*)], \tag{1}$$

where τ is the tree condition number, p_k is the tree-defined potentials for f_k, and

$$\text{gap}(f, p) = \mathcal{E}(f) - (2b^T p - p^T L_G p).$$

The above inequality (1) implies

$$\mathbf{E}[\mathcal{E}(f_{k+1}) - \mathcal{E}(f_*)] \leq \left(1 - \frac{1}{\tau}\right) \mathbf{E}[\mathcal{E}(f_k) - \mathcal{E}(f_*)].$$

We also showed that the initial flow f_0 satisfies

$$\mathcal{E}(f_0) - \mathcal{E}(f_*) \leq (\text{st}_T(G) - 1)\mathcal{E}(f_*).$$

Therefore, after $k = \tau \ln(\text{st}_T(G)\tau/\epsilon)$ iterations,

$$\begin{aligned}
\mathbf{E}[\mathcal{E}(f_k) - \mathcal{E}(f_*)] &\leq \left(1 - \frac{1}{\tau}\right)^k (\text{st}_T(G) - 1)\mathcal{E}(f_*) \\
&\leq \mathrm{e}^{-\ln(\text{st}_T(G)\tau/\epsilon)}(\text{st}_T(G) - 1)\mathcal{E}(f_*) \\
&\leq \frac{\epsilon}{\tau}\mathcal{E}(f_*),
\end{aligned}$$

where the inequality $1 - x \leq \mathrm{e}^{-x}$ is used in the second step.

In the following, we will see how to bound the error between the obtained approximate potentials p_k and the desired potentials $p_* = L_G^+ b$ based on the above bound on energy. Denote $\|x\|_L = \sqrt{x^T L_G x}$. We want to show that

$$\|p_k - p_*\|_L \leq \epsilon \|p_*\|_L,$$

[0]This lecture is based in part on the paper by Kelner, Orecchia, Sidford, and Zhu from 2013, `https://arxiv.org/abs/1301.6628`, and in part by a survey of Arora, Hazan, and Kale 2012 `http://theoryofcomputing.org/articles/v008a006/v008a006.pdf`.

so that the potentials p_k are close to p_* under $\|x\|_L$. Note that $\|p_*\|_L^2 = p_*^T L_G p_* = \mathcal{E}(f_*)$, so we are trying to show that p_k is within ϵ of the energy of the optimal electrical flow f_*. We have that

$$\begin{aligned}\|p_k - p_*\|_L^2 &= \|p_k - L_G^+ b\|_L^2 = (p_k - L_G^+ b)^T L_G (p_k - L_G^+ b) \\ &= p_k^T L_G p_k - 2 p_k^T L_G L_G^+ b + p_*^T L_G p_* \\ &= -(2 p_k^T b - p_k^T L_G p_k) + \mathcal{E}(f_*) = \mathrm{gap}(f_*, p_k).\end{aligned}$$

Observe that (1) also implies

$$\mathbf{E}[\mathrm{gap}(f_k, p_k)] = \tau \mathbf{E}[\mathcal{E}(f_k) - \mathcal{E}(f_{k+1})] \leq \tau \mathbf{E}[\mathcal{E}(f_k) - \mathcal{E}(f_*)].$$

Hence

$$\begin{aligned}\mathbf{E}[\|p_k - p_*\|_L^2] &= \mathbf{E}[\mathrm{gap}(f_*, p_k)] = \mathbf{E}[\mathrm{gap}(f_k, p_k)] - \mathbf{E}[\mathcal{E}(f_k) - \mathcal{E}(f_*)] \\ &\leq (\tau - 1)\mathbf{E}[\mathcal{E}(f_k) - \mathcal{E}(f_*)] \leq \epsilon \mathcal{E}(f_*) \\ &= \epsilon p_*^T L_G p_* = \epsilon \|p_*\|_L^2,\end{aligned}$$

and $\mathbf{E}[\|p_k - p_*\|_L] \leq \sqrt{\epsilon}\|p_*\|_L$.

There are some research questions related to the above algorithm:

- Can this algorithm be made deterministic without changing the running time?
- Can we find a nearly-linear-time Laplacian solver without using low-stretch trees? The paper proposing the above algorithm claimed that the low-stretch tree can be replaced by the Bartal tree, but it did not spell out the details.

2 The Multiplicative Weights Update Algorithm

In this section, we will introduce a very useful algorithm that has been repeatedly discovered by different people in many fields. At first glance, this algorithm seems have no relationship with the main subject of our course. However, in the future lecture, we will demonstrate a fast algorithm for maximum flows combining electrical flows and the idea here.

Assume at each time step, there are N choices for a decision to be made. At time t, we will gain an unknown value $v_t(i) \in [0, 1]$ for making decision i, and the values of all decisions will be revealed after the decision is made. Surprisingly, there is a simple strategy which guarantees to do as well as the best fixed decision over the time.

In the multiplicative weights update algorithm (Algorithm 1), a weight is maintained for each decision i. If we let $w_t(i)$ be weight for decision i at start of time step t and

$$W_t = \sum_{i=1}^{N} w_t(i),$$

then decision i is chosen with probability proportional to $w_t(i)$, so the probability we make decision i at time t is $p_t(i) = w_t(i)/W_t$. After making the decision, the weights will be updated so as to be larger for decisions that get larger values. When the algorithm terminates,

the expected value gained by the algorithm is

$$\sum_{t=1}^{T}\sum_{i=1}^{N} p_t(i)v_t(i).$$

Algorithm 1: Multiplicative Weights

$w_1(i) \leftarrow 1,\ \forall i = 1, \ldots, N$
for $t \leftarrow 1$ **to** T **do**
  Pick i with probability $p_t(i)$, get value $v_t(i)$
  $w_{t+1}(i) \leftarrow (1 + \epsilon v_t(i))w_t(i),\ \forall i = 1, \ldots, N$
end

Theorem 1 *Assume $\epsilon \leq 1/2$, then for any j,*

$$\sum_{t=1}^{T}\sum_{i=1}^{N} p_t(i)v_t(i) \geq (1-\epsilon)\sum_{t=1}^{T} v_t(j) - \frac{1}{\epsilon}\ln N.$$

Proof: The proof idea is to find both the upper and lower bound on W_{T+1}. Note that

$$\begin{aligned} W_{t+1} = \sum_{i=1}^{N} w_{t+1}(i) &= \sum_{i=1}^{N} w_t(i)(1 + \epsilon v_t(i)) \\ &= W_t + \epsilon W_t \sum_{i=1}^{N} p_t(i)v_t(i) \\ &= W_t\left(1 + \epsilon\sum_{i=1}^{N} p_t(i)v_t(i)\right) \\ &\leq W_t \exp\left(\epsilon\sum_{i=1}^{N} p_t(i)v_t(i)\right). \end{aligned}$$

Therefore,

$$W_{T+1} \leq W_1 \exp\left(\epsilon\sum_{t=1}^{T}\sum_{i=1}^{N} p_t(i)v_t(i)\right) = N\exp\left(\epsilon\sum_{t=1}^{T}\sum_{i=1}^{N} p_t(i)v_t(i)\right).$$

On the other hand, for any given j,

$$W_{T+1} \geq w_{T+1}(j) = \prod_{t=1}^{T}(1 + \epsilon v_t(j)) \geq (1+\epsilon)^{\sum_{t=1}^{T} v_t(j)},$$

using the result $1 + \epsilon x \geq (1+\epsilon)^x$ for $x \in [0, 1]$.

Combining the above two inequalities, we get

$$(1+\epsilon)^{\sum_{t=1}^{T} v_t(j)} \leq N \exp\left(\epsilon \sum_{t=1}^{T}\sum_{i=1}^{N} p_t(i)v_t(i)\right).$$

Taking the logarithm of each side,

$$\ln(1+\epsilon)\sum_{t=1}^{T} v_t(j) \leq \ln N + \epsilon \sum_{t=1}^{T}\sum_{i=1}^{N} p_t(i)v_t(i),$$

which implies

$$\sum_{t=1}^{T}\sum_{i=1}^{N} p_t(i)v_t(i) \geq \frac{\ln(1+\epsilon)}{\epsilon}\sum_{t=1}^{T} v_t(j) - \frac{1}{\epsilon}\ln N \geq (1-\epsilon)\sum_{t=1}^{T} v_t(j) - \frac{1}{\epsilon}\ln N.$$

Here in the last step we are using the inequality $\ln(1+x) \geq x - x^2$ for $x \leq 1/2$. □

For the case where each decision i is associated with a cost $c_t(i) \in [-1,1]$ instead of value $v_t(i)$, Algorithm 1 can be modified accordingly to guarantee the expected cost

$$\sum_{t=1}^{T}\sum_{i=1}^{N} p_t(i)c_t(i) \leq \sum_{t=1}^{T}(c_t(j) + \epsilon|c_t(j)|) + \frac{1}{\epsilon}\ln N$$

for each j. For a good overview of multiplicative weights method, see the survey written by Arora, Hazan and Kale.

3 Multiplicative Weights for Packing Problems

In this section, we apply the multiplicative weights method to find feasible solutions to the system:

$$Ax \leq e, \quad x \in Q. \tag{2}$$

Here $A \in \mathbb{R}^{m\times n}$, $e \in \mathbb{R}^m$ is the vector of all ones, and $Q \subseteq \mathbb{R}^n$ is a convex set. Assume $Ax \geq 0$ for each $x \in Q$.

We also assume that it is easy to optimize over Q, i.e., we have an *oracle* which can find $x \in Q$ such that $p^T Ax \leq p^T e$ if such an x exists given any nonnegative vector $p \in \mathbb{R}^m$. If no such $x \in Q$ exists, then we can conclude that the system (2) is infeasible. Since $p^T Ax$ is a linear function in x, we have an oracle as long as we can optimize linear functions over Q.

The goal is to find approximate solution $x \in Q$ to (2) such that $Ax \leq (1+\epsilon)x$, which can be solved by Algorithm 2 based on the multiplicative weights method. For convenience, define the *width* ρ of the oracle to be

$$\rho = \max_{i=1,\ldots,m} \max_{\substack{x\in Q \\ \text{returned} \\ \text{by oracle}}} (Ax)(i).$$

The idea in Algorithm 2 is to run multiplicative weights algorithm in which each decision corresponds to a row of A and its value is $(Ax_t)(i)/\rho$, where x_t is a vector returned by the oracle.

Algorithm 2: Finding Feasible Solution to (2)

$w_1(i) \leftarrow 1,\ \forall i = 1, \ldots, m$
for $t \leftarrow 1$ **to** T **do**
  $W_t \leftarrow \sum_{i=1}^{m} w_t(i), \quad p_t(i) \leftarrow w_t(i)/W_t$
  Run oracle to find $x_t \in Q$ such that $p_t^T A x_t \leq p_t^T e$
  $v_t(i) \leftarrow (Ax_t)(i)/\rho$ (observe the value is in $[0,1]$ by the definition of width ρ)
  $w_{t+1}(i) \leftarrow (1 + \epsilon v_t(i)) w_t(i),\ \forall i = 1, \ldots, m$
end
return $\bar{x} = \sum_{t=1}^{T} x_t/T$

The running time is $O(Tm)$ time plus $O(T)$ oracle calls and additional matrix-vector multiplications. The intuition in Algorithm 2 is to increase the weights most on the most violated inequalities, so in later iterations the oracle will work harder to find x_t satisfying these constraints.

The returned value $\bar{x}$ is always in Q by the convexity of Q. What remains to do is to bound $A\bar{x}$. Observe that

$$\sum_{i=1}^{m} p_t(i) v_t(i) = \frac{1}{\rho} p_t^T A x_t \leq \frac{1}{\rho} p_t^T e = \frac{1}{\rho}.$$

By Theorem 1, for any j,

$$\begin{aligned}
\frac{T}{\rho} &\geq \sum_{t=1}^{T} \sum_{i=1}^{N} p_t(i) v_t(i) \geq (1-\epsilon) \sum_{t=1}^{T} v_t(j) - \frac{1}{\epsilon} \ln m \\
&= (1-\epsilon) \sum_{t=1}^{T} \frac{1}{\rho} (Ax_t)(j) - \frac{1}{\epsilon} \ln m \\
&= (1-\epsilon) \frac{T}{\rho} (A\bar{x})(j) - \frac{1}{\epsilon} \ln m.
\end{aligned}$$

Hence

$$(1-\epsilon) \frac{T}{\rho} (A\bar{x})(j) \leq \frac{T}{\rho} + \frac{1}{\epsilon} \ln m.$$

Set $T = \rho \ln m/\epsilon^2$,

$$(A\bar{x})(j) \leq \frac{1}{1-\epsilon} \left(1 + \frac{\rho \ln m}{\epsilon T}\right) = \frac{1+\epsilon}{1-\epsilon} \leq 1 + 4\epsilon$$

for $\epsilon \leq 1/3$, which gives $A\bar{x} \leq (1+4\epsilon)e$. The running time is

$$O\left(\frac{m\rho}{\epsilon^2} \ln m\right) + O\left(\frac{\rho}{\epsilon^2} \ln m\right) \text{ oracle calls.}$$