ORIE 6334 Spectral Graph Theory October 18, 2016

Lecture 16

Lecturer: David P. Williamson *Scribe: Alberto Vera*

In this lecture, we give a simple, combinatorial algorithm for approximately solving the Laplacian system $L_G p = b$ in nearly linear time. In some sense, the existence of such a solver is the reason this course exists; it was the first such solver, due to Spielman and Teng, that started a set of research that motivated this course and others like it.

1 Algorithm

We can give a sketch of the algorithm below. Recall that, given a tree T and a flow f, the *tree-defined potentials* are $p(r) = 0$ for a selected root vertex r and $p(i) = \sum_{(k,l)\in P(i,r)} r(k,l)f(k,l)$, where $P(i,r)$ is the directed (i,r) path in T. Recall also that any electrical flow obeys both the Kirchoff Current Law (KCL, or flow conservation) and the Kirchoff Potential Law (KPL), which says that $\sum_{(i,j)\in C} r(i,j)f(i,j) = 0$ for any directed cycle C. The algorithm will work by maintaining a flow f that obeys KCL, and will keep picking cycles and fixing the flow so that it obeys KPL on the cycle.

The algorithm is as follows:

Algorithm 1: Simple Combinatorial Laplacian Solver

Find low-stretch tree T
Find flow f_0 in T that satisfies supplies b
Let p_0 be the potentials defined by T and f_0
for $i \leftarrow 1$ **to** $O(st_T(G)\log(st_T(G)/\varepsilon))$ **do**
    Pick $e \in E - T$
    Update f_i to satisfy KPL on cycle formed by adding e to T
    Let f_{i+1}, p_{i+1} be the resulting flow and tree-defined potentials
return p_i

We will use the energy $\mathcal{E}(f) = \sum_{(i,j)\in E} r(i,j)f(i,j)^2$ as a potential function. The following lemma was proved in a previous lecture.

Lemma 1 *For a given b, $b^T e = 0$:*

1. *The electrical flow f minimizes $\mathcal{E}(g)$ among all g that obey flow conservation/KCL.*
2. *The potentials p for electrical flow f maximize $2b^T x - x^T L_G x$ among all $x \in \mathbb{R}^n$. Additionally, $2b^T p - p^T L_G p = \mathcal{E}(f)$.*

We observed at the time that for a flow f and potentials p, these give us an upper and a lower bound on the energy of an electrical flow, and we will use these bounds in the

[0]This lecture is based on a paper by Kelner, Orecchia, Sidford, and Zhu from 2013, `https://arxiv.org/abs/1301.6628`.

algorithm. In particular, for the given resistances r, let f_* be the optimal electrical flow, p_* associated potentials. Then for a flow f that obeys KCL and potentials p, we define the associated gap between the upper and lower bounds given by the lemma as

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

so for any f that obeys KCL, any potentials p, $\mathcal{E}(f) - \mathcal{E}(f_*) \leq \text{gap}(f,p)$.

To prove that the algorithm converges to the optimal electrical flow (approximately), we will ultimately prove two statements. First, we will prove that the energy of the initial flow is not that far away from the energy of the optimal flow, by showing

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

Second, we will show that in each iteration the gap is reduced by a factor of approximately $1 - \frac{1}{\text{st}_T(G)}$. Using these two statements, the flow f obtained after $\text{st}_T(G)\ln(\text{st}_T(G)/\varepsilon)$ iterations is such that

$$\mathcal{E}(f) - \mathcal{E}(f_*) \leq \left(1 - \frac{1}{\text{st}_T(G)}\right)^{\text{st}_T(G)\log(\text{st}_T(G)/\varepsilon)} \text{st}_T(G)\mathcal{E}(f_*) \leq \varepsilon\mathcal{E}(f_*).$$

Of course, what we are really after is a proof that the potentials returned by the algorithm are approximately a solution to $L_G p = b$, but we will show later that coming close to $\mathcal{E}(f_*)$ is useful in finding such a p.

2 Analysis

2.1 Decreasing Energy by Fixing KPL on Cycles

Each iteration of the algorithm is to fix KPL on a cycle. Why does this help? We show that by doing this, the energy decreases. For a directed cycle C, let

$$\Delta(C,f) \equiv \sum_{(i,j)\in C} r(i,j)f(i,j) \quad \text{and} \quad R(C) \equiv \sum_{(i,j)\in C} r(i,j).$$

If $\Delta(C,f) \neq 0$, then let

$$f'(i,j) := \begin{cases} f(i,j) - \varepsilon & \text{if } (i,j) \in C \\ f(i,j) & \text{otherwise.} \end{cases}$$

We observe that by decreasing f by ϵ on each arc around a directed cycle, if KCL was obeyed previously by f, it is still obeyed for f' since the flow into and out of each node i on the cycle has been decreased by ϵ. Setting $\varepsilon = \Delta(C,f)/R(C)$,

$$\Delta(C,f') = \Delta(C,f) - \sum_{(i,j)\in C} r(i,j)\frac{\Delta(C,f)}{R(C)} = 0.$$

The next lemma proves that the energy of the flow in fact decreases by changing the flow to obey KPL on cycle C. Note that if $\Delta(C,f) \neq 0$, then $-\Delta(C,f)^2$ is always negative.

Lemma 2 $\mathcal{E}(f') - \mathcal{E}(f) = -\Delta(C,f)^2/R(C)$.

Proof:

$$\begin{aligned}\mathcal{E}(f') - \mathcal{E}(f) &= \sum_{(i,j)\in C} r(i,j)[(f(i,j)-\varepsilon)^2 - f(i,j)^2] \\ &= \sum_{(i,j)\in C} r(i,j)[\varepsilon^2 - 2\varepsilon f(i,j)] \\ &= \varepsilon^2 R(C) - 2\varepsilon\Delta(C,f).\end{aligned}$$

Since we set $\varepsilon = \Delta(C,f)/R(C)$, the last equation gives the result. □

Recall that we defined $\vec{E}$ to be the set of directed arcs formed by taking the undirected edges in E and giving each an arbitrary orientation. Let $C(i,j)$ be a directed cycle defined by taking $(i,j) \in \vec{E} - T$ and the directed (j,i) path in T. The next lemma shows that the gap between our upper bound and our lower bound is extremely related to the amount by which the energy decreases by fixing KPL on a cycle.

Lemma 3 *For tree-defined potentials p,*

$$\text{gap}(f,p) = \sum_{(i,j)\in\vec{E}-T} \frac{\Delta(C(i,j),f)^2}{r(i,j)}.$$

Proof:

$$\begin{aligned}\text{gap}(f,p) &= \mathcal{E}(f) - [2b^T p - p^T L_G p] \\ &= \sum_{(i,j)\in\vec{E}} r(i,j)f(i,j)^2 - 2\sum_{i\in V} b(i)p(i) + \sum_{(i,j)\in\vec{E}} \frac{(p(i)-p(j))^2}{r(i,j)}.\end{aligned}$$

Observe that, using skew symmetry (that is, that $f(i,j) = -f(j,i)$),

$$\sum_{i\in V} b(i)p(i) = \sum_{i\in V} p(i) \sum_{j:(i,j)\in E} f(i,j) = \sum_{(i,j)\in\vec{E}} f(i,j)(p(i)-p(j)).$$

If we plug this into the expression for the gap,

$$\begin{aligned}\text{gap}(f,p) &= \sum_{(i,j)\in\vec{E}} \Big[r(i,j)f(i,j)^2 - 2f(i,j)(p(i)-p(j)) + \frac{(p(i)-p(j))^2}{r(i,j)} \Big] \\ &= \sum_{(i,j)\in\vec{E}} [r(i,j)f(i,j) - (p(i)-p(j))]^2/r(i,j). \qquad (1)\end{aligned}$$

For $(i,j) \in T$, by definition of the tree-defined potentials, $p(i) - p(j) = r(i,j)f(i,j)$. Hence, $(i,j) \in T$ does not contribute to the sum in (1).

For $(i,j) \in \vec{E} - T$, let $P(i,j)$ be the directed (i,j)-path in T. Since $p(i) - p(j) = \sum_{(k,l)\in P(i,j)} r(k,l)f(k,l)$,

$$r(i,j)f(i,j) - (p(i)-p(j)) = r(i,j)f(i,j) + \sum_{(k,l)\in P(j,i)} r(k,l)f(k,l) = \Delta(C(i,j),f).$$

Using the last expression in (1) finishes the proof. □

2.2 Tree Condition Number

We now take a slight detour to define a useful concept related to the stretch of a tree. The *tree condition number* is defined as

$$\tau(T) \equiv \sum_{(i,j)\in\vec{E}-T} \frac{R(C(i,j))}{r(i,j)}.$$

Since the tree T is fixed throughout the algorithm, we'll just use $\tau \equiv \tau(T)$. Since the stretch of $(i,j) \in T$ is 1 and that of $(i,j) \in \vec{E} - T$ is $\frac{R(C(i,j))-r(i,j)}{r(i,j)}$, we have that

$$\begin{aligned}\tau &= \sum_{(i,j)\in\vec{E}-T} (\mathrm{st}_T(i,j)+1) \\ &= (\mathrm{st}_T(G) - |T|) + (m - |T|) \\ &= \mathrm{st}_T(G) + m - 2(n-1) = \tilde{O}(m),\end{aligned}$$

given that we find a tree of stretch $\tilde{O}(m)$.

2.3 Main Idea

We can now give the main idea at the heart of the algorithm. In the main loop of the algorithm, we need to pick some edge to fix KPL on the induced cycle. We sample edge $(i,j) \in \vec{E} - T$ each iteration with probability $\frac{1}{\tau}\frac{R(C(i,j))}{r(i,j)}$, where $\tau = \tau(T)$. Using the two lemmas we proved, the expected decrease in energy in each iteration is

$$\frac{1}{\tau}\sum_{(i,j)\in\vec{E}-T} \frac{R(C(i,j))}{r(i,j)} \cdot \frac{\Delta(C(i,j),f)^2}{R(C(i,j))} = \frac{1}{\tau}\sum_{(i,j)\in\vec{E}-T} \frac{\Delta(C(i,j))^2}{r(i,j)} = \frac{\mathrm{gap}(f,p)}{\tau}.$$

Hence, we obtain

$$\mathbb{E}[\mathcal{E}(f_i) - \mathcal{E}(f_*)] \leq \left(1 - \frac{1}{\tau}\right)\mathbb{E}[\mathcal{E}(f_{i-1}) - \mathcal{E}(f_*)],$$

and we can decrease the energy by a factor of $(1 - 1/\tau)$ in each iteration.

2.4 Bound on Energy of Initial Flow

We have so far one of the two claimed properties. Now we prove the bound on the energy of the initial flow.

Lemma 4 $\mathcal{E}(f_0) - \mathcal{E}(f_*) \leq (st_T(G) - 1)\mathcal{E}(f_*)$.

Proof: We create a flow in the tree T from the optimal flow f_* in the following way: For each $(k,l) \in \vec{E}$, we send $f_*(k,l)$ units on the (k,l)-path in T, $P(k,l)$. Then for any $(i,j) \in T$ the flow on the edge (i,j) is

$$\sum_{(k,l)\in\vec{E}:(i,j)\in P(k,l)} f_*(k,l).$$

We claim that there is a unique flow in T that satisfies the supply vector b. In particular, if removing the edge (i,j) splits the tree into two sets of vertices S and $V-S$ with $i \in S$, $j \in V-S$, then $f(i,j) = \sum_{k \in S} b(k)$. Thus in particular, the flow defined above must be the same as f_0, and so has the same energy as f_0. So

$$\mathcal{E}(f_0) = \sum_{(i,j)\in T} r(i,j) \left(\sum_{(k,l)\in \vec{E}:(i,j)\in P(k,l)} f_*(k,l) \right)^2.$$

Then, applying Cauchy-Schwarz, we get

$$\begin{aligned}
\mathcal{E}(f_0) &= \sum_{(i,j)\in T} r(i,j) \left(\sum_{(k,l)\in \vec{E}:(i,j)\in P(k,l)} f_*(k,l) \right)^2 \\
&\leq \sum_{(i,j)\in T} \left(\sum_{(k,l)\in \vec{E}:(i,j)\in P(k,l)} \frac{r(i,j)}{r(k,l)} \right) \left(\sum_{(k,l)\in \vec{E}:(i,j)\in P(k,l)} r(k,l) f_*(k,l)^2 \right) \\
&\leq \sum_{(i,j)\in T} \left(\sum_{(k,l)\in \vec{E}:(i,j)\in P(k,l)} \frac{r(i,j)}{r(k,l)} \right) \mathcal{E}(f_*) \\
&= \sum_{(k,l)\in \vec{E}} \left(\sum_{(i,j)\in P(k,l)} \frac{r(i,j)}{r(k,l)} \right) \mathcal{E}(f_*) \\
&= \mathrm{st}_T(G) \cdot \mathcal{E}(f_*).
\end{aligned}$$

Rearranging terms gives the lemma statement. □

Here is an alternate proof (from Renato Paes Leme) that was not presented in class. On the problem set, we have you show that for a spanning tree T, $L_G \preceq \mathrm{st}_T(G)\, L_T$. Renato observes that for potentials p associated with the electrical flow f_0 on the tree T,

$$\begin{aligned}
\mathcal{E}(f_0) = 2b^T p - p^T L_T p &\leq 2b^T p - \frac{1}{\mathrm{st}_T(G)} p^T L_G p \\
&= \mathrm{st}_T(G) \left(2b^T (p/\mathrm{st}_T(G)) - (p/\mathrm{st}_T(G))^T L_G (p/\mathrm{st}_T(G)) \right) \\
&\leq \mathrm{st}_T(G)\, \mathcal{E}(f_*),
\end{aligned}$$

since by Lemma 1 we know that potentials $p/\mathrm{st}_T(G)$ give a lower bound on the optimal energy.

Thus if we run the algorithm for $k = \tau \ln(\mathrm{st}_T(G)\tau/\epsilon)$ iterations, we get that

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

We postpone until next time a discussion why this is sufficient to find potentials that approximately solve $L_G p = b$.

3 Running Time

We now analyze the overall running time. We can find a low-stretch tree in $O(m \log n \log \log n)$ time. It is possible to compute the probability distribution we need in $O(m)$ time; note that it is the same distribution in each step. We are not explaining how to do this, or how to sample an edge from the distribution. Also, in each of the $O(\tau \log(\mathrm{st}_T(G)\tau/\varepsilon))$ iterations, we need to update the flow and potentials. If the tree were a balanced binary tree with r at the root, it is easy to see that we could do update the flow and potentials along the cycle in $O(\log n)$ time; we assert that there exists a data structure that lets our algorithm perform these updates in $O(\log n)$ time.

Thus the total running time is thus $\tilde{O}(m \log(n/\varepsilon))$.

The fastest running time known so far for the problem is

$$O(m\sqrt{\log n} \log \log^{3+\delta} n \log \frac{1}{\epsilon}),$$

for any $\delta > 0$ and is due to Cohen, Kyng, Pachocki, Peng, and Rao from 2014. Note that this is faster than sorting m numbers!