# Lecture 19

## 1 The Laplacian

The next few lectures focus on the problem of solving $Ax = b$ where $A$ is a matrix derived from a graph in nearly linear time. Arbitrary linear systems can be solved in time $O(n^3)$ using Gaussian elimination, but it is possible to do better if $A$ is the Laplacian of a graph.

The Laplacian $L$ of a weighted graph $G$ is the $n \times n$ matrix defined as follows:

$$L_{ij} = \begin{cases} -w_{ij} & \text{if } i \sim j \\ w_i & \text{if } i = j \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

The weight $w_i = \sum_{j \sim i} w_{ij}$ is the sum of the weights of edges incident on vertex $i$. The Laplacian is a positive semidefinite matrix i.e. the eigenvalues of the Laplacian are non negative. Expanding $x$ in the spectral basis, it is easy to see that a symmetric matrix $L$ is positive semidefinite if and only if $x^T Lx \geq 0$ for all $x \in \mathbb{R}^n$.

The Laplacian quadratic form $x^T Lx$ can be evaluated by expressing the Laplacian as a a sum over $2 \times 2$ matrices $L(u, v) = \begin{bmatrix} w_{uv} & -w_{uv} \\ -w_{uv} & w_{uv} \end{bmatrix}$ for $(u, v) \in E(G)$.

$$x^T Lx = \sum_{(u,v) \in E} x^T L(u, v)x = \sum_{(u,v) \in E} w_{uv}(x_u - x_v)^2 \tag{2}$$

The expression shows that the Laplacian is positive semidefinite and if $G$ is connected, the all 1s vector is the unique eigenvector with eigenvalue 0. The Laplacian of a $d$ regular graph is equal to $dI - A$ where $A$ is the adjacency matrix. The eigenvalues are $d - \lambda_i$ where $\lambda_i$ are the eigenvalues of the adjacency matrix $A$.

### 1.0.1 Electrical Networks

Laplacian linear systems arise naturally in the theory of electrical networks. Assume that the network is linear so that Ohm's law $V = IR$ holds, and the edges of the graph all have resistance 1 ohm. The potential difference across an edge is equal to the current through it, so given vertex potentials $v_i$, the outgoing current at vertex $i$ is given by,

$$\sum_{j \sim i} (v_i - v_j) = (Lv)_i \tag{3}$$

Multiplying the potential vector $v$ by the Laplacian yields the vector of outgoing currents $i$. The vertex potentials for a given current vector are computed by solving a Laplacian linear system, here $L^+$ denotes the (pseudo)-inverse of the Lapacian,

$$Lv = i \qquad\qquad L^+ i = v \tag{4}$$

### 1.0.2 Symmetric diagonally dominant matrices

Linear systems in symmetric diagonally dominant ($SDD$) matrices reduce to solving Laplacian linear systems. A $SDD$ matrix $A$ can be uniquely decomposed as $A = L + P$ where $L$ is a Laplacian obtained by selecting the diagonal and negative entries of $A$ while $P$ is a matrix with zeroes on the diagonal and some off diagonal positive entries.

Solving the linear system $Ax = b$ reduces to solving $L'(x, -x) = (b, -b)$ where $L'$ is the Laplacian defined below.

$$\begin{bmatrix} L & -P \\ -P & L \end{bmatrix} \begin{bmatrix} x \\ -x \end{bmatrix} = \begin{bmatrix} Ax \\ -Ax \end{bmatrix} \tag{5}$$

Note that $A$ must be $SDD$ for $L'$ to be a Laplacian, and the solutions to $Ax = b$ can be read off from the first $n$ coordinates of the solution to $L'x = (b, -b)$.

### 1.0.3 Today's lecture

An important component of the fast Laplacian linear system solver is an algorithm for approximating graphs by low stretch spanning trees. We will make the notion of stretch precise and approximate a graph by a tree, though the tree will not be a spanning tree. A spanning tree with total stretch $\widetilde{O}(m \log n)$ can be found in nearly linear time.

## 2 Approximating graphs by trees

### 2.0.4 Metrics:

A metric is a symmetric distance function such that the triangle inequality $d(x, y) \leq d(x, z) + d(z, x)$ holds. A graph $G = (V, E)$ defines a metric on the set of vertices where $d(i, j)$ is the length of the shortest path between $i$ and $j$. We would like to approximate a graph metric by a tree metric, as trees have few edges and are nice from the perspective of algorithm design.

If the distances in the tree metric satisfy $d(i, j) \leq d_T(i, j) \leq \alpha d(i, j)$, then $\alpha$ is called the distortion or stretch for the edge $(i, j)$. The example of a cycle of length $n$ shows that the worst case stretch for a spanning tree can be $n$. However, if we pick a random spanning tree for the cycle, the expected stretch for all edges is 2.

*Distortion lower bound:* A probabilistic embedding of $G$ with distortion $\alpha$ is a specified by a distribution $T$ on trees such that,

$$d(i, j) \leq E_T[d_T(i, j)] \leq \alpha d(i, j) \qquad \forall (i, j) \in E(G) \tag{6}$$

We saw that the average stretch for the cycle was 2, in fact the worst case average stretch can be $O(\log n)$. The optimal spanning tree for the $k \times k$ grid is the $H$ tree which is constructed recursively by joining the four $H$ trees for the $k/2 \times k/2$ grids by an $H$ shaped structure. We will not prove that the $H$ tree has optimal stretch, but will show that the average stretch for the $H$ tree is $O(\log k)$. To contrast with the $H$ tree consider the row tree where all rows of the grid and one column are included, a typical edge has stretch $k$.

There are $2k$ middle edges in the $H$ tree and each of them has a stretch of $2k$, their contribution to the total stretch stretch is $4k^2$. At the next recursive level, there are $4k$

middle edge having stretch $k$ so the contribution to the total stretch is $4k^2$ again. There are $\log n$ levels so the total stretch is at least $4k^2 \log k$, the average stretch being $O(\log k)$.

It is always possible to approximate a graph metric by a tree metric in the sense of (6) with distortion $\alpha = O(\log n)$. This is proved in [**?**] by constructing a hierarchical decomposition for the graph.

## 2.1 Hierarchical decomposition

Let the diameter of the graph be $\Delta$, the number of levels in the hierarchical decomposition will be $\delta = \log \Delta$. The $i$-th level in the decomposition consists of clusters of radius $2^i$, the leaves at level 0 correspond to the $n$ single vertices while level $\delta$ is the single cluster $V$.

The hierarchical clustering algorithm creates a tree decomposition for $G$ as follows:

1. Parameters: A uniformly random permutation $\pi : [n] \to [n]$ that determines an ordering on the vertices, $\beta \in [1, 2]$ chosen uniformly at random.
2. The clustering is built from top to bottom starting with the single cluster $V$ at level $\delta$, to construct level $i-1$ from level $i$ the clusters $D_i$ at level $i$ are decomposed as follows.
3. Let $u$ be an iterator over vertices in order $\pi(1), \pi(2), \cdots, \pi(n)$, create a new cluster at level $i-1$ consisting of unassigned vertices $v \in D_i$ such that $d(u, v) \leq \beta 2^{i-1}$.

Note that the center $u$ of a new cluster need not belong to the cluster and a center can correspond to more than one cluster at a given level. The clustering algorithm yields a tree of depth $\delta$ with vertices for each cluster and edges between a cluster $D_i$ at level $i$ and all its sub-clusters at level $i-1$. The weight of an edge between levels $i$ and $i-1$ in the tree is $2^i$.

The distance $d_T(u, v)$ in the tree metric for vertices $(u, v) \in E(G)$ can be computed as follows: Let $d \leq \delta$ be the smallest level at which the nodes $u$ and $v$ belong to the same cluster, then $d_T(u, v)$ is equal to twice $(1 + 2 + 2^2 + \cdots + 2^d) = 2^{d+1} - 1$ as in this many moves both $u$ and $v$ reach the center of the common cluster. For all edges $(u, v) \in E(G)$ we have,

$$d_T(u, v) < 2^{d+2} d(u, v) \tag{7}$$

The average distortion $\alpha$ can therefore be bounded by computing the expected value of $2^{d+2}$ over the edges of $G$.

Let $w_1, w_2, \cdots, w_n$ be the vertices in increasing order of distance from $u$. The edge $(u, v)$ is separated by a cluster centered at $w_i$ only if $d(w_i, u) = \beta 2^j$ and $d(w_i, v) = \beta 2^j + 1$ for some $j$ and $w_i$ is the first vertex from the ordering $w$ in the random order $\pi$. The expected distance between $u$ and $v$ in the tree metric can be bounded by considering the first center $w_i$ that separates the edge $(u, v)$.

$$\begin{aligned}
E_T[d(u, v)] &= \sum_{i \in [n]} \Pr_{\beta, \pi}[(u, v) \text{ is separated by } w_i] . 2^{\lfloor \log d(u, w_i) \rfloor + 2} \\
&\leq \sum_{i \in [n]} \Pr_{\beta}[d(w_i, u) = \beta 2^{\lfloor \log d(u, w_i) \rfloor}] . \Pr_{\pi}[w_i \text{ occurs first}] . 2^{\lfloor \log d(u, w_i) \rfloor + 2} \\
&= \sum_{i \in [n]} \frac{1}{2^{\lfloor \log d(u, w_i) \rfloor}} \cdot \frac{1}{i} . 2^{\lfloor \log d(u, w_i) \rfloor + 2} = O(\log n)
\end{aligned} \tag{8}$$

# 3 Sparsest cut, oblivious routing

We extend the hierarchical decomposition (a tree where the leaves are vertices of $G$ and internal nodes are vertex clusters) of a graph following [**?**], the result can be used to obtain an $O(\log n)$ approximation to sparsest cut and an $O(\log n)$ competitive oblivious routing scheme.

A decomposition tree $T$ corresponds to a hierarchical clustering of $G$, the leaves of $T$ are vertices of $G$ and interior nodes correspond to cluster centers. An edge in $T$ defines a cut in $G$, the capacity of the edge is the capacity of the corresponding cut. Multi-commodity flows on $G$ and $T$ can be mapped into each other as follows.

A flow on $G$ between $u$ to $v$ maps to a flow on $T$ through the unique path between leaf nodes $u$ and $v$. A flow between a pair of tree nodes corresponds to a flow on $G$ as tree nodes are cluster centers and tree edges can be mapped to arbitrary paths between cluster centers.

THEOREM 1
*A convex combination of decomposition trees $T_i$ can be found efficiently such that if multicommodity flows $f_i$ on $T_i$ have congestion at most 1, then the flow $\sum \lambda_i f_i$ on $G$ has congestion $O(\log n)$.*

The proof of the theorem is an experts like boosting argument applied to the low stretch decomposition tree construction from section 2, for details see [**?**].

## 3.1 Applications:

### 3.1.1 Sparsest cut

If all the edges of $G$ have unit capacity and there is a demand of $d/n$ units between every pairs of vertices, then the sparsity $\phi(S)$ of cut $(S, \overline{S})$ is the ratio of the capacity to the demand,

$$\phi(S) = \frac{n|E(S,\overline{S})|}{d|S||\overline{S}|} = \frac{\sum_{(u,v)\in E(S,\overline{S})} c(u,v)}{\sum_{u\in S, v\in\overline{S}} d(u,v)} \tag{9}$$

Route this multicommodity flow using the tree decomposition given by theorem 1. The congestion on $e \in T_i$ is the ratio of the demand to the capacity for the induced cut $S_e$ and is equal to $\frac{1}{\phi(S_e)}$. Let $L = \max_{e\in T_i} \frac{1}{\phi(S_e)}$ be the maximum congestion over tree edges, and $(S, \overline{S})$ be the corresponding cut. We show that $(S, \overline{S})$ is an $O(\log n)$ approximation to the sparsest cut on $G$.

The flow $f_i$ on $T_i$ has congestion at most $L$ so the flow $\sum_i \lambda_i f_i$ on $G$ has congestion $O(L \log n)$ by theorem 1. The congestion for a flow on $G$ is at least $\frac{1}{\phi(G)}$ so we have the bound $\frac{1}{O(L\log n)} \leq \phi(G)$.

$$\frac{1}{O(L\log n)} \leq \phi(G) \leq \phi(S) = \frac{1}{L}$$

The cut $S$ is therefore a $\log n$ approximation to the sparsest cut problem.

### 3.1.2 Oblivious Routing

An oblivious routing scheme constructs flows between all pairs of vertices $(u, v)$ without knowing the demands in the network. The scheme is said to be $k$-competitive if the congestion for any demand vector is at most $k$ times the optimal congestion.

Consider the oblivious routing scheme defined by the tree decomposition in theorem 1 i.e. route 1 unit of flow between $(u, v)$ with probability $\lambda_i$ use the path defined by $T_i$.

If there is a multicommodity flow with congestion $C$ for the demand vector $x$, then $C \geq \frac{1}{\phi_x(G)} \geq \frac{1}{\phi_x(S)}$, here $\phi_x$ is the conductance with respect to the demand vector. In particular, all the flows on $T_i$ have congestion at most $C$, and the flow $\sum \lambda_i f_i$ on $G$ has congestion $O(C \log n)$ by theorem 1.