CS369M: Algorithms for Modern Massive Data Set Analysis **Lecture 14 - 11/09/2009**

Flow-based Methods for Clustering and Partitioning Graphs and Data

Lecturer: Michael Mahoney *Scribes: Jacob Bien and Ya Xu*

**Unedited notes*

1 Spectral Methods

1. Find an approximation to best cut in G
2. Time takes to compute Fiedler vector "exactly" or "approximately".

- If the graph is really large, can we find approximation to the best cut near by or for a given size? We would like to inherent some of the provably good properties (theorems) or some of the robustness properties of the global methods:
  (1) do what we did with Cheeger's inequality
  (2) with a vector that's good locally.
- Two senses which you might be local:
  (1) find a good cluster near you
  (2) do all computations locally, i.e. depend on size of set/cut returned
- How to get a vector that is good locally:
  (1) Truncate: random walks from localized start node
  (2) Approximate: PageRank computation with local seed vector
  (3) heat kernels

Recall Cheeger's inequality:

Theorem 1.

$$2h_G \geq \lambda_G \geq \frac{\alpha_G^2}{2} \geq \frac{h_G^2}{2}$$

where α_G is the conductance of the best set along the sweep cut.

Fact: There is a strong relationship between $h_G(\phi_G)$ and rate of convergence of a random walk

Two directions:

(1) Let S be the best cut. S is the set of nodes such that $\phi_S = \min_{S' \subset G} \phi_{S'}$

(2) The probability that the random walk will go to a vertex in $\bar{S}$ is ϕ_s. It needs to run $\sim \frac{1}{4\phi_S}$ steps to get 1/4 mass out of S

Partial Converse: (proof can be found in Chung's "Four proofs..." paper)

(1) If ϕ_S is big then every random walk converges "fast".

(2) If the random walk does not converge fast, then by looking at probability distribution, you can get a good cut.

Theorem 2. *Let W be the lazy random walk matrix, then*

$$|W^t(u,s) - \pi(s)| \leq \sqrt{\frac{vol(S)}{d_u}}\,(1-\beta_t/8)^t$$

where β_t is the conductance value found in the best sweep cut found in first t steps.

Theorem 3 ("Cheeger-like")**.**

$$2h_G \geq \lambda_G \geq \frac{\beta_G^2}{8} \geq \frac{h_G^2}{8}$$

where β_G is the min cheeger ratio

Notes: this is algorithmic time - time to compute $p_0,\ p_1,\ \dots,\ p_t = W^t p_0$. Truncated random walk: if $(p_t)_i \leq \xi$, set $(p_t)_i = 0$.

PageRank PageRank is a way to order vertices of large graph. Recall the W matrix. Then with probability α, the random walk jumps to a new node on G, and with $1-\alpha$ it follows W:

$$p = \alpha(\frac{1}{n}, \cdots, \frac{1}{n}) + (1-\alpha)Wp$$

Personalized PageRank Say we are at a starting node s. Let $v = \chi_s$ be the teleporting vector. Then $p = \alpha\chi_s + (1-\alpha)Wp$, which gives $p = \alpha\sum_{t=0}^{\infty}(1-\alpha)^t W^t \chi_s$.

Recall, $\alpha(S) = \{(u,v), u \in S, v \notin S\}$ is the edge boundary and $\delta(S) = \{v, v \in S, (u,v) \in E, u \notin S\}$ is the vertex boundary and $f : V \to \mathbb{R}$ satisfies the Dirichlet boundary conditions if $f(v) = 0\ \forall v \in \delta(S)$.

Point: Laplacian on G also acts on function on G satisfying Dirichlet boundary condition and the same as Laplacian restricted to S.

Definition

$$h_S = \min_{S \subset T} h(T)$$

the local expansion coefficient.

Theorem 4. *Using the personalized PageRank vector,*

$$h_S \geq \lambda_S \geq \frac{\gamma_s}{8\log(\cdots)}$$

where γ_S is the best sweep cut value.

Point Much of the machinery underlying global spectral methods can be made local

- global computation, local cut
- algorithm running time local

2 Flow based graph partitioning

- using network flow ideas to reveal bottlenecks in graph.
- $G = (V, E)$ s is source, t is sink.
- **Goal:** route as much flow as possible.
- max flow = min cut (duality)

Def *Multicommodity flow problem:* Given $k \geq 1$, (s_i, t_i, D_i), goal is to simultaneously route D_i units of flow from s_i to $t_i \forall i$ while respecting capacity constraints.

- Max throughput flow: max amount of flow summed over all commodities.
- Max concurrent flow: max fraction of demand D_i that can be route by flow...

$$\max f \text{ s.t. } fD_i \text{ units of flow go from } s_i \text{ to } t_i.$$

- $$\text{min cut} = \rho = \min_{U \subseteq V} \frac{C(U, \bar{U})}{D(U, \bar{U})}$$

  where

$$C(U, \bar{U}) = \sum_{e \in (U, \bar{U})} C(e)$$
$$D(U, \bar{U}) = \sum_{\substack{i: s_i \in U \\ t_i \in \bar{U} \text{or v.v.}}} D_i$$

- UMFP: all demands are uniform $\rightarrow$ expansion
- PMFP: $\pi : V \rightarrow R^+$. Demands are $\pi(v_i)\pi(v_j)$. E.g. if $\pi(v) = deg(v) \rightarrow$ conductance.

Fact 1 max-flow/min-cut gap $\leq O(\log k)$ (comes from metric embedding)

Fact 2 If certain conditions are satisfied, then gap=0. Look at dual polytope.
Optimal solution – integral or not.

Fact 3 Worst case (over input graph) gap $\Omega(\log k)$.

Proof. on expanders. Structure of proof like that seen earlier. □

2.1 Algorithmic Applications

UMFP: $D(U, \bar{U}) = |U||\bar{U}|$.

$$\begin{aligned} \text{min cut:} \quad \rho &= \min_{U \subseteq V} \frac{C(U, \bar{U})}{|U||\bar{U}|} \\ &= \min_{U \subseteq V} \frac{E(U, \bar{U})}{|U||\bar{U}|} \quad \text{if all capacities} = 1. \end{aligned}$$

So sparsest cut $\sim$ best expansion.

- "poly-time" – can solve "balanced" cut problem and use it for divide and conquer.
- best running time $O(n^2)$

Aside A local improvement algorithm:

- Goal: Given a partition, find a strictly better partition.
- METIS – post process with a flow based improvement heuristic.
- Vanilla spectral: post process with improvement method.
- Local improvement at one step online iterative algorithm.

Theorem: $A \subseteq V$ s.t. $\pi(A) \leq \pi(\bar{A})$. $S = Improve(A)$ [partition flow algorithm].

1. if $C \subseteq A$, then $Q(S) \leq Q(C)$ [where $Q(S) = |\partial S|/vol(S)$]
2. if C is such that

$$\frac{\pi(A \cap C)}{\pi(C)} \geq \frac{\pi(A)}{\pi(V)} + \epsilon \frac{\pi(\bar{A})}{\pi(V)},$$

   i.e. C is ϵ more correlated with A than random,
   then $Q(S) \leq Q(C)/\epsilon$ i.e. bound on nearby cuts.

- Spectral: relaxation to vector space $O(\log n)$, graph partition.
- Flow: relaxation to l_1 (that's an LP) $O(\log n)$, graph partitioning algorithm.