Stat260/CS294: Spectral Graph Methods **Lecture 10 - 02/24/2015**

Lecture: Flow-based Methods for Partitioning Graphs (1 of 2)

Lecturer: Michael Mahoney *Scribe: Michael Mahoney*

Warning: these notes are still very rough. They provide more details on what we discussed in class, but there may still be some errors, incomplete/imprecise statements, etc. in them.

10 Introduction to flow-based methods

Last time, we described the properties of expander graphs and showed that they have several "extremal" properties. Before that, we described a vanilla spectral partitioning algorithms, which led to the statement and proof of Cheeger's Inequality. Recall that one direction viewed λ_2 as a relaxation of the conductance or expansion problem; while the other direction gave a "quadratic" bound as well as a constructive proof of a graph partitioning algorithm. The basic idea was to compute a vector, show that it is a relaxation of the original problem, and show that one doesn't loose too much in the process. Later, we will see that there are nice connections between these methods and low-dimensional spaces and hypothesized manifolds.

Lest one think that this is the only way to compute partitions, we turn now to a *very* different method to partition graphs—it is based on the ideas of single-commodity and multi-commodity flows. It is *not* a spectral method, but it is important to know about for spectral methods (e.g., when/why spectral methods work and how to diagnose things when they don't work as one expects): the reason is basically that flow-based graph algorithms "succeed" and "fail" in very different ways than spectral methods; and the reason for this is that they too implicitly involve embedding the input graph in a metric/geometric place, but one which is very different than the line/clique that spectral methods implicitly embed the input into.

10.1 Some high-level comments on spectral versus flow

Recall that the key idea underlying graph partitioning algorithms is take a graph $G = (V, E)$ and spread out the vertices V in some abstract space while *not* spreading out edges E too much, and then to partition the vertices in that space into tow (or more) sets.

- **Spectral methods** do this by putting nodes on an eigenvector and then partitioning based on a sweep cut over the partitions defined by that eigenvector. For spectral methods, several summary points are worth noting.
  - They achieve "quadratic" worst-case guarantees from Cheeger's Inequality. This quadratic factor is "real," e.g., it is not an artifact of the analysis and there are graphs on which it is achieved.
  - They are "good" when graph has high conductance or expansion, *i.e.*, when it is a good expander (in either the degree-weighted or degree-unweighted sense).

  - They are associated with some sort of underlying geometry, e.g., as defined by where the nodes get mapped to in the leading eigenvector; but it is not really a metric embedding (or at least not a "good" metric embedding since it only preserves average distances—which is typical of these ℓ_2-based methods—and not the distance between every pair of nodes.

- **Flow-based methods** do this by using multi-commodity flows to reveal bottlenecks in the graph and then partitioning based on those bottlenecks. To contrast with spectral methods, note the following summary points for flow-based methods.

  - They achieve $O(\log n)$ worst-case guarantees. This $O(\log n)$ is "real," in that it too is not an artifact of the analysis and there are graphs on which it is achieved.
  - They are "good" when the graph has sparse cuts, *i.e.*, when it is not a good expander.
  - They are also associated with a geometry, but one which is very different than that of spectral methods. In particular, although not immediately-obvious, they can be viewed as embedding a finite metric space of graph into ℓ_1 metric, *i.e.*, a very different metric than before, and then partitioning there.

The point here is that these two methods are in many ways complementary, in the sense that they succeed and fail in different places. Relatedly, while "real" data might not be exactly one of the idealized places where spectral or flow-based methods succeed or fail, a lot of graph-based data have some sort of low-dimensional structure, and a lot of graph-based data are sufficiently noisy that it is fruitful to view them as having expander-like properties.

Finally, the comments about spectral methods being good on expanders and flow methods being good on non-expanders might seem strange. After all, most people who use spectral methods are not particularly interested in partitioning graphs that do not have good partitions (and instead they take advantage of results that show that spectral methods find good partitions in graphs that are "morally" low-dimensional, e.g., graphs like bounded-degree planar graphs and well-shaped meshes). The reason for this comment is that the metric we have been using to evaluate cluster quality is the objective function and not the quality of the clusters. By this measure, the quadratic of a constant (which is the expansion value of an expander) is a constant, while the quadratic of, e.g., $1/\sqrt{n}$ is very large, i.e., the spectral guarantee is nontrivial in one case and not in the other. For flow-based methods by contrast, the $O(\log n)$ is much larger than the constant value of an expander and so gives trivial results, while this is much smaller than, e.g., $1/\sqrt{n}$, leading to nontrivial results on the objective for morally low-dimensional graphs. That these qualitative guides are the opposite of what is commonly done in practice is a real challenge for theory, and it is a topic we will return to later.

10.2 Flow-based graph partitioning

Recall the basic ideas about flow and the single commodity flow problem:

- $G = (V, E)$ is a graph, and each edge $e \in E$ has capacity $C(e)$, which is the maximum amount of flow allowed through it. Also, we are given a source s and a sink t.

- (We are going to be applying it to graph partitioning by trying to use network flow ideas to reveal bottlenecks in the graph, and then cut there.)

- So, the goal is to route as much flow $s \to t$ without violating any of the constraints.
- Max-flow is the maximum amount of such flow.
- Min-Cut = the minimum amount of capacity to be removed from a network to disconnect s from t; that is,
$$Min - Cut = \min_{U \subset V : s \in U, t \in \bar{U}} \sum_{e \in (U, \bar{U})} c(e).$$
(Note that there is no ratio here, since we can assume that demand = 1, WLOG, but that we won't be able to do this later.)
- Weak duality is one thing and is relatively easy to show:

  Claim 1. $maxflow \leq mincut$

  Proof. For all $U \subseteq V$ that has s and t on opposite sides, all flows from $s \to t$ must be routed through edges in $(U, \bar{U})$, so the total flow is bounded by the capacity in mincut. □
- Strong duality is another thing and is harder:

  Claim 2. $maxflow = mincut$

  Proof. Suffices to show that mincut bound is always achievable is harder. We won't do it here. □

All of this discussion so far is for $k = 1$ single-commodity flow.

There are also multi-commodity versions of this flow/cut problem that have been widely studied. Here is the basic definition.

Definition 1. *Given $k \geq 1$, each with a source s_i and sink t_i and also given demands D_i for each, i.e., we have (s_i, t_i, D_i) for each $i \in [k]$, the* multi-commodity flow problem *is to simultaneously route D_i units of flow from s_i to t_i, for all i, while respecting capacity constraints,* i.e.*, s.t. the total amount of all commodities passing through any edge $\leq$ capacity of that edge. There are several variants, including:*

- Max throughput flow*: maximize the amount of flow summed over all commodities.*
- Max concurrent flow*: specify a demand D_i for each commodity, and then maximize the* fraction *of demand D_i that can be simultaneously shipped or routed by flow:*
$$\max f \text{ s.t. } fD_i \text{ units of flow go from } s_i \text{ to } t_i.$$
i.e.*, the maximum f s.t. fD_i units of capacity i are simultaneously routed without violating the capacity constraints.*

That is, for this multicommodity flow problem, if the flow of commodity i along edge (u, v) is $f_i(u, v)$, then an *assignment of flow* satisfies:

- Capacity constraints.

$$\sum_{i=1}^{k} |f_i(u,v)| \leq c(u,v)$$

- Flow conservation

$$\sum_{w \in V} f_i(u,w) = 0 \quad \text{when } u \neq s_i, t_i$$
$$\text{For all } u, v \quad f_i(u,v) = -f_i(v,u)$$

- Demand satisfaction.

$$\sum_{w \in V} f_i(s_i,w) = \sum_{w \in V} f_i(w,t) = D_i$$

Then the goal is to find a flow that maximizes one of the above variants of the problem.

We can also define a related cut problem.

Definition 2. *The* MinCut *or* Sparsest Cut Ξ*—of an undirected multicommodity flow problem—is the mincut over all cuts of the ratio of the capacity of cut to the demand of the cut,* i.e.*,*

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

where

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

That is, $C(U,\bar{U})$ is the sum of capacities linking U to $\bar{U}$; and $D(U,\bar{U})$ is the sum of demands with source and sinks on opposite sides of the $(U,\bar{U})$ cut.

Finally, we point out the following two variants (since they will map to the expansion and conductance objectives when we consider the application of multi-commodity flow to graph partitioning). There are, of course, other variants of flow problems that we don't consider that are of interest if one is primarily interested in flow problems.

Definition 3. *The* Uniform Multi-Commodity Flow Problem (UMFP)*: here, there is a demand for every pair of nodes, and the demand for every commodity is the same,* i.e.*, they are uniform, WLOG we take to equal* 1*. The* Product Multicommodity Flow Problem (PMFP)*: here, there is a nonnegative weight $\pi(\cdot)$, for all nodes $u \in V$,* i.e.*, $\pi : V \to R^+$. The weights of demands between a pair of nodes u and v are $\pi(v_i)\pi(v_j)$.*

Here are several comments about the last definition.

- UMFP is a special case of PMFP with $\pi(i) = 1$.

- If $\pi(i) = 1$, then we will get a way to approximate the expansion objective.
- If $\pi(v) = deg(v)$, then we will get a way to approximate the conductance objective.

The MaxFlow-MinCut for single commodity is nice since it relates two fundamental graph theoretic entities (that are in some sense dual) via the min-max theorem. But, prior to LR, very little was known about the relationship between MaxFlow and MinCut in this more general multi-commodity flow setting for general graphs. For certain special graphs, it was known that there was an equality, i.e., a zero duality gap; but for general graphs, it is only known that MaxFlow is within a fraction k of the MinCut. This results can be obtained since each commodity can be optimized separately in the obvious trivial way by using $\frac{1}{k}$ of the capacity of the edges—this might be ok for $k = \Theta(1)$, but it is is clearly bad if $k \sim n$ or $k \sim n^2$. Somewhat more technically, if we consider the LP formulation of the Max Multicommodity Flow Problem, then we can make the following observations.

- The dual of this is the LP relaxation of the Min Multicut Problem, *i.e.*, the optimal integral solution to the dual is the Mun Multicut.
- In general, the vertices of the *dual polytope* are NOT integral.
- But, for single commodity flow, they are integral, and so MaxFlow-MinCut theorem is a consequence of LP duality.
- For certain special graphs, it can be shown that they are integral, in which case one has zero duality gap for those graphs.
- Thus, for multicommodity: MaxFlow = MinFractional, *i.e.*, relaxed, Multicut.

Here are several facts that we spend some time discussion:

Fact 1 If we have k commodities, then one can show that max-flow/min-cut gap $\leq O(\log k)$. This can be shown directly, or it can be shown more generally via metric embedding methods.

Fact 2 If certain conditions are satisfied, then the duality gap = 0. If one look at dual polytope, then whether or not this is the case depend on whether the optimal solution is integral or not. This, in turn, depends on special structure of the input.

Fact 3 For k commodities, LR showed that the worst case (over input graph) gap $\Omega(\log k)$. LLR interpreted this geometrically in terms of embeddings. The worst case is achieved, and it is achieved on expanders.

Here, we will spend some time showing these results directly. Then, next time we will describe it more generally from the metric embedding perspective, since that will highlight better the similarities and differences with spectral methods.

10.3 Duality gap properties for flow-based methods

Here, we will show that there is a non-zero duality gap of size $\Theta(\log n)$. We will do it in two steps: first, by illustrating a particular graph (any expander) that has a gap at least $\Omega(\log n)$; and second, by showing that the gap is $O(\log n)$, i.e., is never worse than that.

Let's start by showing that there is a graph for which the duality gap is nonzero by showing a graph for which it is at least $\Theta(\log n)$.

Theorem 1 (LR). *$\forall n, \exists$ an n-node UMFP with MaxFlow f and MinCut Ξ s.t.*

$$f \leq O\left(\frac{\Xi}{\log n}\right)$$

Proof. Consider any graph with certain expansion properties. In particular, let G be a 3-regular n-node graph with unit edge capacities s.t.

$$C(U, \bar{U}) = |(U, \bar{U})| \geq C_{onst} \min\{|U|, |\bar{U}|\}, \quad \forall U, V$$

i.e., an expander. (Such graphs exist by Margoulis, zig-zag, etc. constructions; and moreover a randomly-selected 3-regular graph satisfies this w.h.p.) The first claim is that:

$$\begin{aligned} \Xi &= \min_{U \subseteq V} \frac{(U, \bar{U})|}{|U| \cdot |\bar{U}|} \\ &\geq \min_{U \subseteq V} \frac{C_{onst}}{\max\{|U|, |\bar{U}|\}} \\ &= \frac{C_{onst}}{n-1}. \end{aligned}$$

The second claim is that:

Claim 3. *The MaxFlow for UMFP is $\leq \frac{6}{(n-1)(\log n - 1)}$, which is $\Theta(\log n)$ smaller than MinCut.*

Proof of claim. Since the graph is 3-regular, $\exists \leq \frac{n}{2}$ nodes within a distance $\log n - 3$ of $v \in V$. So, for at least half of the $\binom{n}{2}$ commodities, the shortest path connecting s_i and t_i has at least $\log n - 2$ edges. To sustain a flow of f for such a commodity, at least $f(\log n - 2)$ capacity must be used by commodity. To sustain a flow f, $\forall \binom{n}{2}$ commodities, the capacity of the network must be $\geq \frac{1}{2}\binom{n}{2} f(\log n - 2)$. □

Since the graph is 3-regular with unit capacity, the total capacity is $\leq \frac{3n}{2}$. So,

$$\frac{1}{2}\binom{n}{2} f(\log n - 2) \leq CAPACITY \leq \frac{3n}{2}.$$

So,

$$\begin{aligned} f &\leq \frac{3n}{\binom{n}{2}(\log n - 2)} \\ &= \frac{6}{(n-1)(\log n - 2)} \\ &\leq \frac{6\Xi}{C_{onst}(\log n - 2)} \quad \text{since } \Xi \geq \frac{C_{onst}}{n-1} \\ &= O\left(\frac{\Xi}{\log n}\right). \end{aligned}$$

That is, the MaxFlow for the UMFP on an expander G is $\geq \Theta(\log n)$ factor smaller than the MinCut. □

An expander has diameter $O(\log n)$, and so for expanders, the gap can't be worse than $\Theta(\log n)$, but the following shows that this is true more generally.

Theorem 2 (LR)**.** *The MinCut of UMFP can never be more than* $\Theta(\log n)$ *factor larger than MaxFlow,* i.e.*,*

$$\Omega\left(\frac{\Xi}{\log n}\right) \leq f \leq \Xi.$$

Proof. We can do this by showing a polynomial-time algorithm that finds a cut $(U, \bar{U})$ s.t.

$$\frac{C(U,\bar{U})}{|U| \cdot |\bar{U}|} \leq O(f(\log n))$$

where the LHS is the *ratio cost* of $(U, \bar{U})$; and recall that $\Xi = \min_{S \subseteq V} \frac{C(U,\bar{U})}{|U| \cdot |\bar{U}|}$. The algorithm is based on LP Dual—the dual of Multicommodity Flow is the problem of assigning a fixed weight (where the weights are thought of as distances) to edges of g s.t. one maximizes the cumulative distance between source and sink pairs. (We won't actually go through this now since there is a more elegant and enlightening formulation.) □

In the UMFP, the demand across cut $(U, \bar{U})$ is given by: $D(U, \bar{U}) = |U||\bar{U}|$. So, in particular, the mincut is given by:

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

where $C(U, \bar{U}) = \sum_{e \in (U,\bar{U})} C(e)$.

This is, if the demands are uniform and all the capacities are equal to one, then from UMFP we get our old friend, the sparsest cut $\sim$ best expansion. Among other things, this implies that the $O(\log n)$ approximation for general MultiCommodity Flow in "inherited" by the algorithm for the sparsest cuts problem, and many other related problems as well. In particular, one way to use flow is the following: check all 2^n cuts, and use the single-commodity zero-duality gap result to show that we can take the one with the best single cwommodity flow to get the one with the best mincut. What these results say is that, instead, we can consider the all-pairs multi-commodity flow problem and check a lot less and get a result that is only a little worse.

10.4 Algorithmic Applications

Let's go back to our old friend the *sparsest cut problem*, and here we will make explicit connections with flow based graph partitioning by viewing it from an optimization perspective. This will in turn provide us with an algorithm (that is mentioned in the proof of the above theorem) to solve the problem.

Recall that in the sparsest cut problem, we are given: a graph $G = (V, E)$; a cost function $c(e), \forall e \in E$, *i.e.*, $c : E \to \mathbb{R}^+$; and k pairs of nodes/vertices (s_i, t_i). We will write the problem as an Integer Program (IP). To do so,

- Let $x(e)$, $e \in E$, where $x(e) \in \{0, 1\}$ to indicate if an edge e is cut;
- let $y(i)$, $i \in [k]$, where $y(i) \in \{0, 1\}$ to indicate if commodity i is cut, *i.e.*, is disconnected by this cut; and
- let $\mathcal{P}_i$, $i = 1, 2, \ldots, k$ be the set of paths between s_i and t_i.

Then, what we want is to solve:

$$\begin{aligned}
&\min \frac{\sum_{e \in E} c(e)x(e)}{\sum_{i=1}^{k} d(i)y(i)} \\
&\text{s.t.} \quad \sum_{e \in P} x(e) \geq y(i), \quad \forall P \in \mathcal{P}_i, \quad \forall i = 1, 2, \ldots, k \\
&\qquad y(i) \in \{0, 1\}, \quad i \in [k] \\
&\qquad x(e) \in \{0, 1\}, \quad e \in E
\end{aligned}$$

We want to consider the relaxation of this to the case where replace the last two constraints by $y(i) \geq 0$ and $x(e) \geq 0$ In doing so, note that if (x, y) is a feasible fractional solution, then $(\alpha x, \alpha y)$ is also a feasible fractional solution with the same objective function value. So, WLOG, we can choose the normalization α s.t. $\sum_{i=1}^{k} d(i)y(i) = 1$ to get the following LP:

$$\begin{aligned}
&\min \sum_{e \in E} c(e)x(e) \\
&\text{s.t.} \quad \sum_{i=1}^{k} d(i)y(i) = 1 \\
&\qquad \sum_{e \in P} x(e) \geq y(i), \quad \forall P \in \mathcal{P}_i, \quad \forall i = 1, 2, \ldots, k \\
&\qquad y(i) \geq 0 \quad , x(e) \geq 0.
\end{aligned}$$

Below, we will show that we can compute a cut with sparsity ratio within a factor of $O(\log k)$ of this optimal fractional solution.

BTW, before we do that, let's write the LP dual:

$$\begin{aligned}
&\max \quad \alpha \\
&\text{s.t.} \quad \sum_{p \in \mathcal{P}_i} f(p) \geq \alpha d(i), \quad \forall i \in [k] \\
&\qquad \sum_{i=1}^{k} \sum_{p \in \mathcal{P}_i(e)} f(p) \leq c(e), \quad \forall e \in E \\
&\qquad f(p) \geq 0, \quad \forall P \in \mathcal{P}_i, i \in [k].
\end{aligned}$$

This is the Max Concurrent Flow problem, and $O(\log k)$ approximation gives an *approximate MaxFlow-MinCut Theorem*.

So, to solve this sparsest cut problem, our strategy will be:

- Solve the LP (either the primal or the dual).
- Round the solution to an integral value.

Thus, there are some similarities with spectral—we first solve something to get a real-valued solution, and then we have to "round" to get an integral solution, and the ball game will be to show that we don't loose too much. Of course, solving the LP will *implicitly* embed us in a very different place than solving an eigenvector problem, so we will expect to see different artifactual properties between the two approximation algorithms.

The above discussion gives algorithms that run in polynomial time:

- Off the shelf LP (due to a connection with LP).
- Algorithms for approximately optimal flows of distance functions (*i.e.*, take advantage of the structure of the LP).
- Fastest update of sparsest cut algorithm is $\tilde{O}(n^2)$, with Benczur-Karger sparsification.
- Standard algorithm that runs in something like $\tilde{O}(n^{3/2})$ with push-relabel methods.
- Finally, there is a lot of work on using Laplacian-based solvers to do better, and we may return to these later.

Important Note: The way you would actually solve this in practice is to use some sort of push-relabel code, which is relatively fast, as opposed to the general LP procedure just described, which is easier to analyze theoretically.

10.5 Flow Improve

Here is an aside that with luck we will get back to later. These algorithms have a running time that large enough that it can be challenging to apply to very large graphs—e.g., $O(n^{3/2})$ or especially $O(n^2)$ is certainly too large for "massive" graphs. (Implementations of faster algorithms are still very much a topic of research.) A question arises, can we do something like "local spectral" (which, recall, consisted roughly of a few steps of a random walk), to do a local flow improvement?

The answer is YES—and here is it. The so-called `Improve` algorithm of AL as well as the related MQI method: this algorithm is useful by itself for improving partitions from, say, Metis or some other very fast procedures; and it is useful as a way to speed up spectral and/or as one way to combine spectral-based algorithms with flow-based algorithms. In more detail, the goal of a local improvement algorithm is: Given a partition, find a strictly better partition. A local improvement algorithm is useful in the following contexts:

- METIS – post process with a flow based improvement heuristic.

- Vanilla spectral: post process with improvement method.
- Local improvement at one step online iterative algorithm.

MQI and Improve essentially construct and solve a sequence of s-t MinCut problems on a modified quotient cut objective to add and remove vertices from a proposed cut. (We won't describe them in detail now.) Here is the basic theorem stating their running time and approximation quality bounds.

Theorem 3. *Let $A \subseteq V$ s.t. $\pi(A) \leq \pi(\bar{A})$, and let $S = \textsc{Improve}(A)$ be the output of the Flow Improve Algorithm. Then*

1. *If $C \subseteq A$, (*i.e., *$\forall C \subseteq A$) then $Q(S) \leq Q(C)$ (where $Q(S) = \frac{|\partial S|}{Vol(S)}$)*
2. *If C is such that*

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

   i.e., *if C is ϵ-more-correlated with A than random,* i.e., *if the fraction of C that is in A is ϵ-better than random, then $Q(S) \leq \frac{1}{\epsilon} Q(C)$* i.e., *bound on nearby cuts.*
3. *The algorithm takes time: (1) $\pi(V)^2$ iterations if vertex weights $\pi(V) \in \mathcal{Z}$; and (2) m iterations if the edges are unweighted.*