Lecture 16

In which we define a multi-commodity flow problem, and we see that its dual is the relaxation of a useful graph partitioning problem. The relaxation can be rounded to yield an approximate graph partitioning algorithm.

1 Generalizations of the Maximum Flow Problem

An advantage of writing the maximum flow problem as a linear program, as we did in the past lecture, is that we can consider variations of the maximum flow problem in which we add extra constraints on the flow and, as long as the extra constraints are linear, we are guaranteed that we still have a polynomial time solvable problem. (Because we can still write the problem as a linear program, and we can solve linear programming in polynomial time.)

Certain variants of maximum flow are also easily reducible to the standard maximum flow problem, and so they are solvable using the combinatorial algorithms that we have discussed.

Example 1 (Vertex Capacities) *An interesting variant of the maximum flow problem is the one in which, in addition to having a capacity $c(u,v)$ for every edge, we also have a capacity $c(u)$ for every vertex, and a flow $f(\cdot,\cdot)$ is feasible only if, in addition to the conservation constraints and the edge capacity constraints, it also satisfies the vertex capacity constraints*

$$\sum_{u:(u,v)\in E} f(u,v) \leq c(v) \quad \forall v \in V$$

It is easy to see that the problem can be reduced to the standard maximum flow problem, by splitting every vertex v into two vertices v_{in} and v_{out}, adding one edge (v_{in}, v_{out}) of capacity $c(v)$, and then converting every edge (u,v) to an edge (u, v_{in}) and every edge (v,w) to an edge (v_{out}, w). It is easy to show that solving the (standard) maximum flow problem on the new network is equivalent to solving the maximum flow with vertex capacity constraints in the original network.

Example 2 (Multiple Sources and Sinks and "Sum" Cost Function) *Several important variants of the maximum flow problems involve multiple source-sink pairs $(s_1, t_1), \ldots, (s_k, t_k)$, rather than just one source and one sink. Assuming that the "stuff" that the sources want to send to the sinks is of the same type, the problem is to find multiple feasible flows $f^1(\cdot,\cdot)$, ..., $f^k(\cdot,\cdot)$, where $f^i(\cdot,\cdot)$ is a feasible flow from the source s_i to the sink t_i, and such that the capacity constraints*

$$\sum_{i=1}^{k} f^i(u,v) \leq c(u,v) \quad \forall (u,v) \in E$$

are satisfied. Such a flow is called a "multi-commodity" flow.

How do we measure how "good" is a multicommodity flow? A simple measure is to consider the sum of the costs

$$\sum_{i=1}^{k} \sum_{v} f^i(s_i, v)$$

In such a case, we can do a reduction to the standard maximum flow problem by adding a "super-source" node s, connected with edges of infinite capacity to the sources s_i, and a "super-sink" node t, to which all sinks t_i are connected to, via infinite capacity edges. It is easy to see that the maximum flow from s to t is the same as the maximum sum of flows in a feasible multicommodity flow in the original network.

In many applications, looking at the sum of the costs of the various flows $f^i(\cdot,\cdot)$ is not a "fair" measure. For example, if the underlying network is a communication network, and $(s_1, t_1), (s_2, t_2)$ are pairs of nodes that need to communicate, a solution that provides $5Mb/s$ of bandwidth between s_1 and t_1 and no bandwidth between s_2 and t_2 is not a very good solution compared, for example, with a solution that provides $2Mb/s$ of bandwidth each between s_1 and t_1 and between s_2 and t_2. (Especially so from the point of view of s_2 and t_2.) There are various reasonable measures of the quality of a multicommodity flow which are more fair, for example we may be interested in maximizing the median flow, or the minimum flow. A rather general problem, which can be used to find multicommodity flows maximizing various cost measures is the following.

Definition 3 (Multicommodity Feasibility Problem) *Given in input a network $G = (V, E_G)$ with capacities $c(u,v)$ for each $(u,v) \in E_G$, and given a collection of (not necessarily disjoint) pairs (s_1, t_1), ..., (s_k, t_k), each having a demand $d(s_i, t_i)$, find a feasible multicommodity flow $f^1(\cdot,\cdot), \ldots, f^k(\cdot,\cdot)$ such that*

$$\sum_{v} f^i(s_i, v) \geq d(s_i, t_i) \quad \forall i = 1, \ldots, k$$

or determine that no such multicommodity flow exists.

A more general version, which is defined as an optimization problem, is as follows.

Definition 4 (Maximizing Fractional Demands) *Given in input a network $G = (V, E_G)$ with capacities $c(u, v)$ for each $(u, v) \in E_G$, and given a collection of (not necessarily disjoint) pairs (s_1, t_1), ..., (s_k, t_k), each having a demand $d(s_i, t_i)$, find a feasible multicommodity flow $f^1(\cdot, \cdot), \ldots, f^k(\cdot, \cdot)$ such that*

$$\sum_v f^i(s_i, v) \geq y \cdot d(s_i, t_i) \quad \forall i = 1, \ldots, k$$

and such that y is maximized.

Note that the vertices $s_1, \ldots, s_k, t_1, \ldots, t_k$ need not be distinct. For example, in the case of a communication network, we could have a broadcast problem in which a node s wants to send data to all other nodes, in which case the source-sink pairs are all of the form (s, v) for $v \in V - \{s\}$. It is useful to think of the pairs of vertices that require communication as defining a weighted graph, with the weights given by the demands. We will call $H = (V, E_H)$ the graph of demands. (In the broadcast example, H would be a star graph.)

The Fractional Multicommodity Flow Problem can be easily formulated as a linear program.

$$\begin{array}{lll}
\text{maximize} & y & \\
\text{subject to} & & \\
& \sum_u f^{s,t}(u, v) = \sum_w f^{s,t}(v, w) & \forall (s, t) \in E_H \forall v \in V - \{s, t\} \\
& \sum_{(s,t) \in E_H} f^{s,t}(u, v) \leq c(u, v) & \forall (u, v) \in E_G \\
& \sum_v f^{s,t}(s, v) \geq y \cdot d(s, t) & \forall (s, t) \in E_H \\
& f^{s,t}(u, v) \geq 0 & \forall (s, t) \in E_H, (u, v) \in E_G
\end{array} \tag{1}$$

As for the standard maximum flow problem, it is also possible to give a formulation that involves an exponential number of variables, but for which it is easier to derive the dual.

In the following formulation, $P_{s,t}$ is the set of all paths from s to t in G, and we have a variable x_p for each path in $P_{s,t}$, for each $(s, t) \in E_H$, corresponding to how many

units of flow from s to t are routed through the path p.

$$\begin{array}{lll} \text{maximize} & y & \\ \text{subject to} & & \\ & \displaystyle\sum_{p \in P_{s,t}} x_p \geq y \cdot d(s,t) & \forall (s,t) \in E_H \\ & \displaystyle\sum_{p:(u,v)\in p} x_p \leq c(u,v) & \forall (u,v) \in E_G \\ & x_p \geq 0 & \forall p \\ & y \geq 0 & \end{array} \tag{2}$$

Note that if E_H contains only one edge (s,t), and $d(s,t) = 1$, then we have the standard maximum flow problem.

2 The Dual of the Fractional Multicommodity Flow Problem

The dual of (2) has one variable $w(s,t)$ for each $(s,t) \in E_H$, and a one variable $z(u,v)$ for each $(u,v) \in E_G$. It is as follows:

$$\begin{array}{lll} \text{minimize} & \sum_{u,v} z(u,v)c(u,v) & \\ \text{subject to} & & \\ & \displaystyle\sum_{(s,t)\in E_H} w(s,t)d(s,t) \geq 1 & \\ & -w(s,t) + \displaystyle\sum_{(u,v)\in p} z(u,v) \geq 0 & \forall (s,t) \in E_H, p \in P_{s,t} \\ & w(s,t) \geq 0 & \forall (s,t) \in E_H \\ & z(u,v) \geq 0 & \forall (u,v) \in E_G \end{array} \tag{3}$$

Thinking a bit about (3) makes us realize that, in an optimal solution, without loss of generality $w(s,t)$ is be the shortest path from s to t in the graph weighted by the $z(u,v)$. Indeed, the constraints force $w(s,t)$ to be *at most* the length of the shortest $z(\cdot,\cdot)$-weighted path from s to t, and, if some $w(s,t)$ is strictly smaller than the length of the shortest path, we can make it equal to the length of the shortest path without sacrificing feasibility and without changing the cost of the solution. The other observation is that, in an optimal solution, we have $\sum w(s,t)d(s,t) = 1$, because, in a solution in which $\sum w(s,t)d(s,t) = c > 1$, we can divide all the $w(s,t)$ and all the $z(u,v)$ by c, and obtain a solution that is still feasible and has smaller cost. This means that the following linear program is equivalent to (3). We have a

variable $\ell(x, y)$ for every pair of vertices in $E_G \cup E_H$:

$$\begin{array}{ll} \text{minimize} & \sum_{u,v} \ell(u,v)c(u,v) \\ \text{subject to} & \\ & \displaystyle\sum_{(s,t)\in E_H} \ell(s,t)d(s,t) = 1 \\ & \displaystyle\sum_{(u,v)\in p} \ell(u,v) \geq \ell(s,t) \qquad \forall (s,t) \in E_H, p \in P_{s,t} \\ & \ell(u,v) \geq 0 \qquad\qquad\qquad\quad \forall (u,v) \in E_G \cup E_H \end{array} \tag{4}$$

3 The Sparsest Cut Problem

From now on, we restrict ourselves to the case in which the graphs E_G and E_H are undirected. In such a case, we have a variable $\ell(u, v)$ for each *unordered* pair u, v. The constraints $\sum_{(u,v)\in p} \ell(u, v) \geq \ell(s, t)$ can be equivalently restated as the *triangle inequalities*

$$\ell(u_1, u_3) \leq \ell(u_1, u_2) + \ell(u_2, u_3)$$

This means that we are requiring $\ell(u, v)$ to be non-negative, symmetric and to satisfy the triangle inequality, and so it is a *metric* over V. (Technically, it is a *semimetric* because we can have distinct vertices at distance zero, and $\ell(\cdot, \cdot)$ is not defined for all pairs, but only for pairs in $E_G \cup E_H$, although we could extend it to all pairs by computing all-pairs shortest paths based on the weights $\ell(x, y)$ for $(x, y) \in E_G \cup E_H$.)

These observations give us one more alternative formulation:

$$\min_{\ell(\cdot,\cdot) \text{ metric}} \frac{\sum_{(u,v)\in E_G} c(u,v) \cdot \ell(u,v)}{\sum_{(s,t)\in E_H} d(s,t) \cdot \ell(s,t)}$$

Now, finally, we can see that the above formulation is the linear programming relaxation of a *cut* problem.

If $A \subseteq V$ is a subset of vertices, we say that a pair (u, v) is *cut* by A if $u \in A$ and $v \notin A$, or vice versa.

Given an instance of the multicommodity flow problem, we say that a subset A of vertices is a *cut* if it cuts at least one of the pairs in E_H. The *sparsest cut* (also called quotient cut) problem is to find a cut A that minimizes the ratio

$$\frac{\sum_{(u,v)\in E_G \text{ cut by } A} c(u,v)}{\sum_{(s,t)\in E_H \text{ cut by } A} d(s,t)}$$

which is called the *sparsity* of the cut A.

Note that, if E_H contains only one pair (s,t), and $d(s,t) = 1$, then we have exactly the standard minimum cut problem.

Suppose that, in our multicommodity problem, there is a fractional flow of cost y. Then, for each pair (s,t) that is cut by A, the $yd(s,t)$ units of flow from s to t must pass through edges of E_G that are also cut by A. Overall, $\sum_{(s,t)\text{cut by } A} yd(s,t)$ units of flow must pass through those edges, whose overall capacity is at most $\sum_{(u,v)\text{cut by } A} c(u,v)$, so we must have

$$\sum_{(u,v)\text{cut by } A} c(u,v) \geq y \sum_{(s,t)\text{cut by } A} d(s,t)$$

which means that the sparsity of A must be at least y. This means that sparsity of every cut is at least the fractional cost of any flow. (This is not surprising because we derived the sparsest cut problem from the dual of the flow problem, but there is a very simple direct reason why the above bound holds.)

Now it would be very nice if we had an exact rounding algorithm to find the optimum of the sparsest cut problem.

For a given graph $G = (V, E_G)$ with capacities $c(u,v)$, if we define E_H to be a clique and $d(s,t) = 1$ for all s, t, then solving the sparsest cut problem on G and H becomes the problem of finding a set A that minimizes

$$\frac{\sum_{(u,v)\in E_G \text{ cut by } A} c(u,v)}{|A| \cdot |V - A|}$$

and optimizing such a cost function tends to favor finding sets A that are large and that have few edges coming out. This is useful in a number of contexts. In clustering problems, if the capacities represent similarity, a sparsest cut algorithm will pick out sets of vertices that are mostly similar to each other, but dissimilar to the other vertices, that is, a cluster. Very effective image segmentation algorithms are based on applying sparsest cut approximation algorithms (but not the one we are describing in these notes, which is too slow) to graphs in which there is a vertex for every pixel, and edges connect nearby pixels with a capacity corresponding to how likely the pixels are to belong to same object in the image.

Unfortunately, the sparsest cut problem is NP-hard. Rounding (4), however, it is possible to achieve a $O(\log |E_H|)$-factor approximation.

We very briefly describe what the approximation algorithm looks like.

First, we need the following result:

Lemma 5 *For every input G, H, c, d, and every feasible solution $\ell(\cdot,\cdot)$ of (4), it is possible to find in polynomial time a subset S of vertices, such that if we define*

$$g_S(v) := \min_{a \in S} \ell(a, v)$$

then we have

$$\sum_{(s,t)\in E_H} \ell(s,t)d(s,t) \leq O(\log |E_H|) \sum_{(s,t)\in E_H} |g_S(s) - g_S(t)|d(s,t)$$

The proof is rather complicated, and we will skip it.

Then we have the following fact:

Lemma 6 *For every input G, H, c, d, every feasible solution $\ell(\cdot,\cdot)$ of (4), and every subset S of vertices, if we define $g_S(v) := \min_{a\in S} \ell(a, v)$, we have*

$$\sum_{(u,v)\in E_G} \ell(u,v)c(u,v) \geq \sum_{(u,v)\in E_G} |g_S(u) - g_S(v)|c(u,v)$$

PROOF: It is enough to show that we have, for every u, v,

$$\ell(u,v) \geq |g_S(u) - g_S(v)|$$

Let a be the vertex such that $\ell(a,u) = g_S(u)$ and b be the vertex such that $\ell(b,v) = g_S(v)$. (They need not be different.) Then, from the triangle inequality, we get

$$\ell(u,v) \geq \ell(u,b) - \ell(b,v) \geq \ell(u,a) - \ell(b,v) = g_S(u) - g_S(v)$$

and

$$\ell(u,v) \geq \ell(v,a) - \ell(a,u) \geq \ell(v,b) - \ell(u,a) = g_S(v) - g_S(a)$$

and so

$$\ell(u,v) \geq |g_S(u) - g_S(v)|$$

□

Lemma 7 *For every input G, H, c, d, and every function $g : V \to \mathbb{R}$, we can find in polynomial time a cut A such that*

$$\frac{\sum_{(u,v)\in E_G \text{ cut by } A} c(u,v)}{\sum_{(s,t)\in E_H \text{ cut by } A} d(s,t)} \leq \frac{\sum_{(u,v)\in E_G} |g(u)-g(v)|c(u,v)}{\sum_{(s,t)\in E_H} |g(s)-g(t)|d(s,t)}$$

PROOF: We sort the vertices in ascending value of g, so that we have an ordering $u_1, \ldots, u_n$ of the vertices such that

$$g(u_1) \leq g(u_2) \leq \cdots \leq g(u_n)$$

We are going to consider all the cuts of the form $A := \{u_1, \ldots, u_k\}$, and we will show that at least one of them has sparsity at most

$$r := \frac{\sum_{(u,v)\in E_G} |g(u)-g(v)|c(u,v)}{\sum_{(s,t)\in E_H} |g(s)-g(t)|d(s,t)}$$

Since r does not change if we scale $g(\cdot)$ by a multiplicative constant, we will assume without loss of generality that $g(u_n) - g(u_1) = 1$.

Let us pick a threshold T uniformly at random in the interval $[g(u_1), g(u_n)]$, and define the set $A := \{u : g(u) \leq T\}$. Now note that, for every pair of vertices x, y, the probability that (x, y) is cut by A is precisely $|g(x) - g(y)|$, and so

$$\mathbb{E} \sum_{(u,v)\in E_G \text{ cut by } A} c(u,v) = \sum_{(u,v)\in E_G} |g(u)-g(v)|c(u,v)$$

and

$$\mathbb{E} \sum_{(s,t)\in E_H \text{ cut by } A} d(s,t) = \sum_{(s,t)\in E_H} |g(s)-g(t)|d(s,t)$$

so that

$$\mathbb{E} \sum_{(u,v)\in E_G \text{ cut by } A} c(u,v) - r \sum_{(s,t)\in E_H \text{ cut by } A} d(s,t) = 0$$

and so there must exist an A in our sample space (which consists of sets of the form $\{u_1, \ldots, u_k\}$) such that

$$\sum_{(u,v)\in E_G \text{ cut by } A} c(u,v) - r \sum_{(s,t)\in E_H \text{ cut by } A} d(s,t) \geq 0$$

that is,

$$\frac{\sum_{(u,v)\in E_G \text{ cut by } A} c(u,v)}{\sum_{(s,t)\in E_H \text{ cut by } A} d(s,t)} \leq r$$

□

This is enough to have an $O(\log |E_H|)$-approximate algorithm for the sparsest cut problem.

On input the graphs G, H, the capacities $c(\cdot,\cdot)$ and the demands $d(\cdot,\cdot)$, we solve the linear program (4), and find an optimal solution $\ell(\cdot,\cdot)$ of cost *optlp*. Then we use Lemma 5 to find a set S such that, if we define $g_S(v) := \min_{a \in S} \ell(a, v)$, we have (using also Lemma 6)

$$\frac{\sum_{(u,v)\in E_G} |g(u) - g(v)| c(u,v)}{\sum_{(s,t)\in E_H} |g(s) - g(t)| d(s,t)} \leq optlp \cdot O(\log |E_H|)$$

Finally, we use the algorithm of Lemma 7 to find a cut A whose sparsity is at most $optlp \cdot O(\log |E_H|)$, which is at most $O(\log |E_H|)$ times the sparsity of the optimal cut. This proves the main result of this lecture.

Theorem 8 *There is a polynomial time $O(\log |E_H|)$-approximate algorithm for the sparsest cut problem.*