CS 683: Advanced Algorithms **Lecturer: Prof. Jon Kleinberg**
Lecture 3, January 30, 2001 **Scribes: Elliot Anshelevich, Alexandre Evfimievski**[1]

1 Fractional Packing vs. Fractional Hitting Set

Here and thereafter, for any two vectors $\mathbf{a}$ and $\mathbf{b}$ such that $\dim \mathbf{a} = \dim \mathbf{b} = n$,

$$\mathbf{a} \leqslant \mathbf{b} \;\Leftrightarrow\; \forall\, i = 1 \ldots n :\; a_i \leqslant b_i.$$

Definition 1.1. Fractional Packing *is the following optimization problem:*

$$\mathbf{1}^T \cdot \mathbf{x} \to \max \;\; \textit{conditioned by} \;\; A\mathbf{x} \leqslant \mathbf{1} \;\textit{and}\; \mathbf{x} \geqslant \mathbf{0}, \tag{1}$$

where $\mathbf{0} = (0, 0, \ldots, 0)^T$ *and* $\mathbf{1} = (1, 1, \ldots, 1)^T$.

Definition 1.2. Fractional Hitting Set *is the following optimization problem:*

$$\mathbf{1}^T \cdot \mathbf{y} \to \min \;\; \textit{conditioned by} \;\; \mathbf{y}A \geqslant \mathbf{1} \;\textit{and}\; \mathbf{x} \geqslant \mathbf{0}. \tag{2}$$

These are the natural generalizations of Packing and Hitting Set problems considered in the previous lecture. A is a 0–1 matrix where columns give membership relations for each set. The condition $A\mathbf{x} \leqslant \mathbf{1}$ in (1) corresponds to the requirement that no element (of the sets) should be overloaded.

The problem Fractional Hitting Set is the dual of Fractional Packing in terms of linear programming. It is a covering problem: "Each set has at least one unit of value. Minimize the value". Here are the intuitive formulations of both optimization problems:

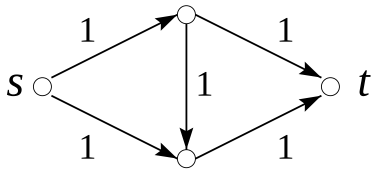
Fractional Packing: "Put a weight on each set so that the weight of all sets containing any given element is at *most* 1. Maximize total weight."

Fractional Hitting Set: "Put a nonnegative value on each element so that the total value of all elements in each set is at *least* 1. Minimize total value."

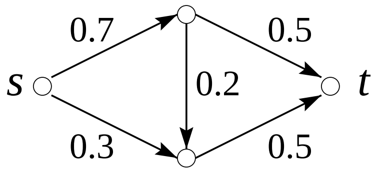
2 Maximum Flow and Linear Programming

We assume that you are familiar with the definition of Maximum Flow and its basic properties from CS 681. Now we are going to look at Maximum Flow on graph $G = (V, E)$ from the viewpoint of linear programming. Consider the special case where all edge weights (capacities) are equal to 1:

[1]Questions? Please e-mail a scribe at `aevf@cs.cornell.edu`



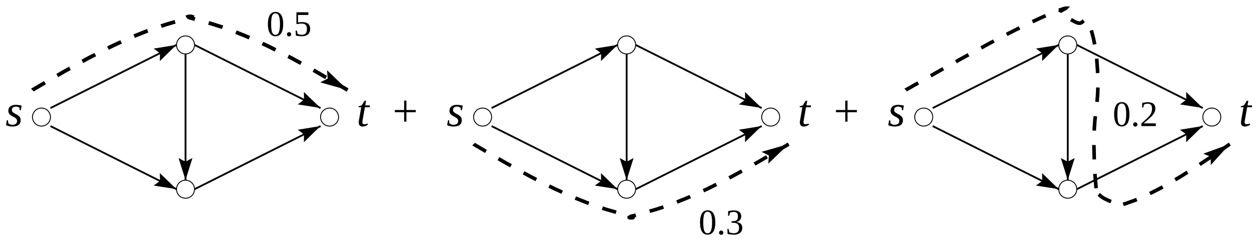
A flow on such graph (a non-maximal one) may look, for example, as follows; here each edge is marked with the flow amount going through it:



From CS 681 some of you may remember the so-called "path decomposition of flows". Given a flow like above, we arbitrarily choose a path P from vertex s to vertex t with non-zero flow f_P running through it. Then we find an edge e on this path such that the amount of flow f_e running through e equals f_P:

$$e \in P : f_e = f_P \stackrel{\text{def}}{=} \min_{e \in P} f_e.$$

Now we can subtract the entire flow running through P, thus leaving edge e without any flow. By iterating this procedure we decompose the flow into paths:



In order to formulate Maximum Flow problem in terms of linear program, we have to consider as variables not just all flow amounts $\{f_e\}$ for all edges, but for all paths P from s to t. With the $\{f_e\}$'s we would have the following setting:

$$\sum_{e \text{ out of } s}^{e \in E} f_e - \sum_{e \text{ into } s}^{e \in E} f_e \to \max, \quad \textit{conditioned by}$$

$$\forall v \in V,\ v \neq s, t : \sum_{e \text{ out of } v}^{e \in E} f_e - \sum_{e \text{ into } v}^{e \in E} f_e = 0 \quad \textit{and} \quad \forall e \in E : 0 \leqslant f_e \leqslant 1.$$

Alternately, we could represent the maximum flow problem as a standard Fractional Packing problem using flow decomposition, that is, as a sum of partial flows x_P for *every* path P from s to t. Obviously, for one flow there could be many possible flow decompositions.

The maximization problem is now stated as follows:

$$\sum_{P \text{ going } s \to t} x_P \;\to\; \max, \quad \textit{conditioned by} \;\; \forall e \in E : \sum_{P:\, e \in P} x_P \;\leqslant 1 \;\; \textit{and} \;\; \forall P : x_P \geqslant 0.$$

This is a Packing problem, so we can write the dual problem: "Put a value $y_e \geqslant 0$ in each edge $e \in E$, so that for all P going from s to t the total value d_P of all edges in P is at least 1, and $\sum_{e \in E} y_e$ is minimized." The $\{y_e\}$'s can be considered edge lengths, and then $d_P = \sum_{e \in P} y_e$ is the length of path P. However, the dual is not exactly Minimum Cut, as one might expect; instead, it is something like Fractional Minimum Cut.

Let us define the distance $d(s, v)$ between the source s and any vertex $v \in V$ in terms of the $\{y_e\}$'s:

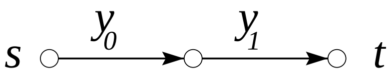
$$d(s, v) \;:=\; \min_{P \text{ going } s \to v} \sum_{e \in P} y_e.$$

In other words, $d(s, v)$ is the distance of the shortest path between s and v. We know from above that, for t, $d(s, t)$ must be at least 1. Duality tells us:

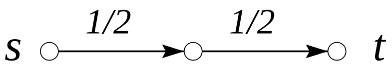
$$\text{Max Flow} \;=\; \sum_{P \text{ going } s \to t} x_P \;=\; \sum_{e \in E} y_e \;=\; \text{Min Length Assignment},$$

but we don't know whether they are also equal to minimum cut.

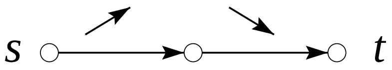
We know, though, that a true minimum cut is greater than or equal to a minimum-length y_e-assignment, because, having a cut, one can easily construct a valid y_e-assignment: just set $y_e = 1$ if edge e goes through the cut (from the s-part to the t-part), and $y_e = 0$ otherwise. Clearly, here any path from s to t is of length at least 1. What if there is no minimum-length y_e-assignment that looks like a cut? Consider a simple example of a y_e-assignment:



What we would like to have here is a 0–1 (or 1–0) assignment, but instead we can get a fractional assignment:



To work around the problem, we would like to "perturb" the assignment so that it becomes non-fractional (the y_e's are rounded up somehow) but still retains its minimality:



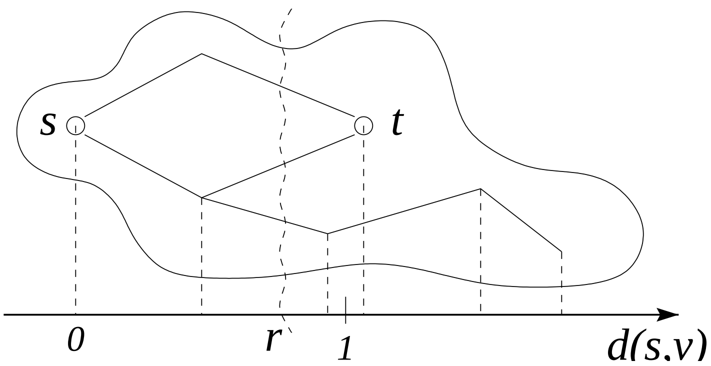
How to round up the y_e's? There are several ideas coming from the field of randomized algorithms:

1-st idea: Treat the y_e's as probabilities (round each y_e to 1 with probability y_e and to 0 with probability $1 - y_e$);

2-nd idea: Need some (easy to analyze) way of correlating these random choices;

3-rd idea: Or, maybe we can prove that there is a positive probability of a valid cut.

But the actual solution is a different rounding procedure. First, project all vertices of the graph onto a line according to their distances $d(s, v)$ from the source:



Then choose r uniformly at random in the interval $(0, 1)$, and round up the y_e's as follows. If $e = (v, w)$ such that $d(s, v) < r$ and $d(s, w) \geqslant r$, then set $y'_e = 1$ and include e in the cut; otherwise set $y'_e = 0$ and don't include in the cut. So, the cut is defined as (A, B) where

$$A := \{v \mid d(s, v) < r\}, \quad B := V \setminus A.$$

Clearly, $s \in A$ and $t \in B$; let us compute the capacity c of such a cut.

The capacity c is a random variable that depends on r. By its definition,

$$c = \sum_{e \in E} X_e, \quad \text{where } X_e := \begin{cases} 1 & \text{if } e \text{ crosses the cut} \\ 0 & \text{otherwise} \end{cases}$$

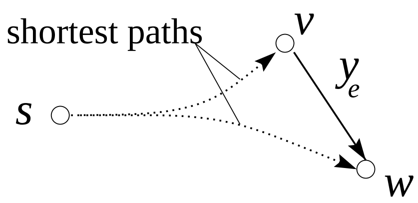
It follows that

$$\mathsf{E}\, c = \sum_{e \in E} \mathsf{E}\, X_e = \sum_{e \in E} \mathsf{P}[X_e = 1].$$

What is $\mathsf{P}[X_e = 1]$? If we knew that $\mathsf{P}[X_e = 1] \leqslant y_e$, then $\mathsf{E}\, c \leqslant \sum_{e \in E} y_e$ which is the optimal length sum, so we would have proven that there exists a cut with the optimal y_e-assignment (it cannot have better than optimal assignment), and, therefore, that $\sum_{e \in E} y_e$ is the optimal cut capacity. So, let us show that it is true:

$$\mathsf{P}[X_e = 1] = \mathsf{P}[\, d(s, v) \leqslant r < d(s, w)\,],$$

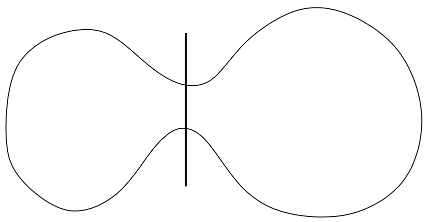
and $d(s, w) - d(s, v) \leqslant y_e$, because otherwise edge (v, w) short-cuts the shortest path from s to w:



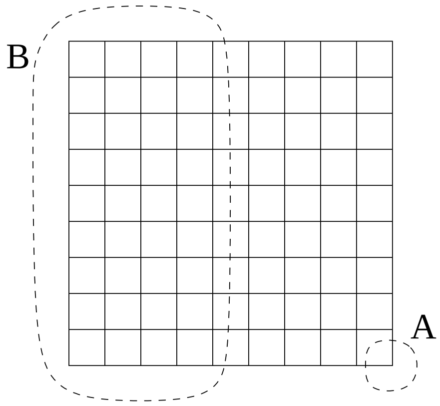
Actually, from the above reasoning it follows that *all* cuts produced like this (by choosing r) are optimal. The reasoning is similar to the proof that if Y is a random variable, $Y \geqslant 0$, and $\mathsf{E}\, Y = 0$, then $Y \overset{\text{a.s.}}{\equiv} 0$.

3 Graph Partitioning

Here, and in the next lecture, we shall consider the problem of graph partitioning. In the problem of Minimum Cut, we had a source s and a sink t, and we wanted to partition our graph by a cut so that s and t are in the opposite partitions. Now, however, there is no s or t. We want an arbitrary but good-quality partitioning:



A good-quality means that the sizes of both parts of the graph are large with respect to the number of edges between the parts. In the picture below, partition A is bad, whereas the partition B is good:



There are many ways to make this precise. One is by introducing the *surface-to-volume ratio* $\alpha(G)$, which is also called the *expansion of* G, *conductance of* G, *flux* etc.:

$$\alpha(G) := \min \left\{ \frac{e(U, V \setminus U)}{|U|} \;\middle|\; U \subset V,\ |U| \leqslant \frac{|V|}{2} \right\},$$

where $e(U, V \setminus U)$ is the number of edges between U and $V \setminus U$. The graph G is now undirected.

The problem of finding $\alpha(G)$ and the corresponding cut is NP-complete. It also can be generalized to weighted edges and nodes.

Examples: for a chain (a line segment with n vertices on it), $\alpha = 2/n$; for a clique (a graph where any pair of vertices is connected), $\alpha = \Omega(n)$; for a two-dimensional grid like on the picture above, with $n = |V|$ being an even perfect square ($n = (2m)^2$), $\alpha = \frac{2}{\sqrt{n}}$ (this is not obvious), and the optimal cut is cut B (see the picture).

Definition 3.1. *A family of expander graphs is an infinite collection $\{G_i\}$ of graphs with absolute constants d and ε so that all G_i have maximum degree at most d and expansion at least ε.*

Expander graphs are not easy to construct, but they exist and are very important for computer science. More about them is in further lectures. Next time: a natural type of flow for graph partitioning.