CS 683: Advanced Algorithms	Min-Ratio Cuts and Max Concurrent Flow
1 February 2001	Scribes: Aaron Archer and Tom Wexler

1 Solvability of Linear Programs

Thus far we have used linear programs (LP's) as a proof tool but have not addressed how to solve them efficiently. For most of the course we will use LP's as a black box and at the end we will discuss polynomial time algorithms.

Given a number γ, an $m \times n$ matrix A, column vector b, and row vector c, all with integer or rational coefficients, the LP decision problem asks whether there exists a column vector x such that

$$\begin{aligned} Ax &\leq b \\ cx &\geq \gamma. \end{aligned}$$

For this problem to be in NP, it is sufficient that for every "yes" instance there exists a rational column vector x that satisfies $Ax \leq b$ and $cx \geq \gamma$ and such that the number of bits necessary to represent x is polynomial in m, n, and the number of bits in A, b, and c. Using elementary polyhedral theory and Cramer's rule it is not difficult to show in this way that linear programming is in NP, but we will not prove this here. Similarly, to prove we have a "no" instance we can exhibit a dual solution y with polynomial bit complexity. Therefore, linear programming is in co-NP as well.

It is conjectured that all problems residing in both NP and co-NP are also in P. However, a polynomial time algorithm for linear programming eluded researchers for many years, until Khachiyan's ellipsoid algorithm (1978), which we will cover in the last week of class. While this algorithm has many applications to theory, it is unfortunately not practical. Interior point algorithms, beginning with Karmarkar's (1984), are both practical and have polynomial worst case running times. A peculiar feature of these algorithms is that the number of arithmetic operations depends on the number of bits in the input. It remains an open question whether there exists a so called *strongly polynomial* algorithm, where the number of arithmetic operations is polynomial in m and n, independent of the bit complexity.

2 Cuts and Multicommodity Flows

Last time we defined the *expansion* of G to be

$$\alpha(G) = \min_{\substack{U \subseteq V \\ |U| \leq \frac{n}{2}}} \frac{e(U, V-U)}{|U|},$$

where $e(A, B)$ denotes the number of edges from A to B in G.

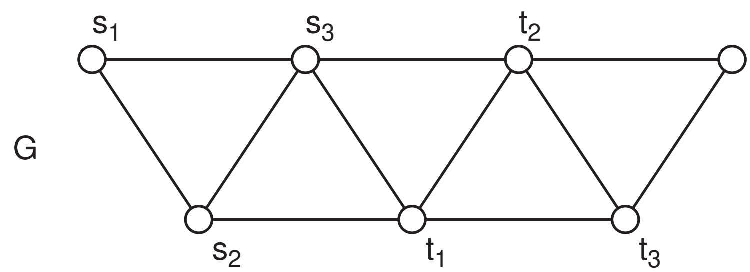

Figure 1: A multicommodity flow network with terminals.

This is strongly related to a similar notion, the *min ratio cut* of G, defined as

$$\sigma(G) = \min_{U \subseteq V} \frac{e(U, V-U)}{|U||V-U|}.$$

Notice that $\frac{\alpha(G)}{\sigma(G)}$ always lies between $\frac{n}{2}$ and n, so that determining one determines the other, up to a factor of 2. Both of these quantities identify a set of nodes with a small "surface area to volume ratio."

As we have seen, cuts and flows are dual objects, where flows are solutions of a packing LP and cuts correspond to integer solutions of the dual covering LP. What kinds of flows are dual to the cuts we have just defined?

Consider the following *multicommodity flow* (MCF) problem. Given an undirected graph G and pairs of vertices $(s_1, t_1), \ldots, (s_k, t_k)$ called *terminals*, are there flows $f_1, \ldots, f_k$ such that f_i sends one unit of flow from s_i to t_i and the total flow on any edge is at most 1? We could generalize this by specifying a demand d_i for each (s_i, t_i) pair and allowing a capacity c_e for each edge in G, but for now we will take d_i and c_e to be 1. Note that for any $i \neq j$, f_i and f_j represent the flow of different goods, so flows in opposite directions do not cancel. That is, c_e represents the capacity for total flow in both directions. We will first consider MCF combinatorially before examining the LP formulation.

2.1 What are the obstacles to multicommodity flow?

Clearly if there exists a multicommodity flow then for all $U \subseteq V$ we have

$$\text{demand across the cut } (U, V-U) \leq e(U, V-U),$$

where the demand across $(U, V-U)$ is the number of (s_i, t_i) pairs split by the cut.

For example, in Figure 2 there is no solution since a feasible flow would need to send two units of flow over edge e.

For ordinary single commodity maximum flow, this cut condition is also sufficient (by the max flow min cut theorem). This is not true for MCF, as demonstrated by Figure 3. Here we can check all cases to show that the cut condition is satisfied. However, there is no

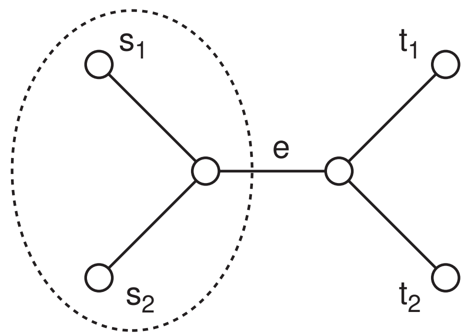

Figure 2: The circled nodes in this MCF network denote a cut proving infeasibility.

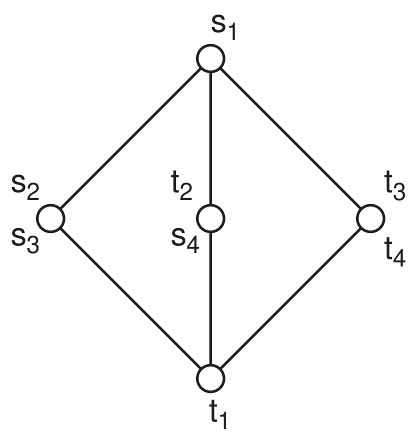

Figure 3: This MCF problem is infeasible even though there is no cut proving so.

feasible MCF. To see this, notice that every commodity must travel across at least two edges to reach its destination. Let us think of flows and edge capacities as cross-sectional areas, and the edges as pipes of length 1. A flow satisfying all demands would eat up 8 units of volume in the system, which has only 6 edges total. This proves that the most flow we could hope to send simultaneously is 3/4 of a unit between each (s_i, t_i) pair. This is essentially a duality argument where the dual variable (length) assigned to each edge is the same.

2.2 Integrality of flows

In MCF, there can be a valid fractional flow where there is no integer-valued flow. See Figure 4 for an example. The pictured fractional flow is feasible, whereas any integer solution would have to use some edge twice. This lack of an integrality theorem is not surprising since the problem of integer flows with unit capacities and demands precisely describes disjoint path problems, which are NP-hard, while fractional MCF can be formulated as a polynomially-sized LP and hence solved in polynomial time.

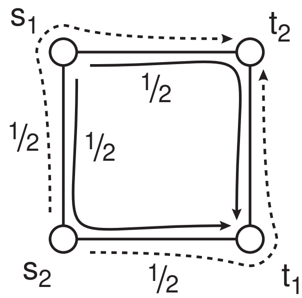

Figure 4: This MCF has a valid fractional flow but no integral flow.

3 Approximate max flow min cut theorems

With respect to a given MCF problem, let us define the *narrowest cut* as

$$\eta(G, \{st\}) = \min_{U \subseteq V} \frac{e(U, V-U)}{\text{number of } (s_i, t_i) \text{ pairs split by } (U, V-U)},$$

where $\{st\}$ denotes the set of terminal pairs. (Note that the η notation is not standard, and we will not necessarily use it in future lectures.) To have a valid flow, we clearly must scale down all demands from 1 to $\leq \eta(G, \{st\})$, and maybe even lower.

Definition 3.1 *Let $\lambda(G, \{st\})$ denote the maximum demand we can give to each (s_i, t_i) pair so as to have a valid MCF.*

The problem of finding $\lambda(G, \{st\})$ and a flow that achieves it is called the *maximum concurrent flow* problem. The next claim just restates the comments above.

Claim 3.2 *For all G, $\{st\}$ we have $\lambda(G, \{st\}) \leq \eta(G, \{st\})$.*

This is a weak duality theorem. The example of Figure 3 shows there is no corresponding strong duality theorem, as we had for linear programming. But this raises the question of whether we can obtain an approximate version, i.e. whether there is some "margin of safety" $c > 1$ (a constant, or perhaps a function of some parameters of the problem) such that

$$\lambda(G, \{st\}) \geq \frac{\eta(G, \{st\})}{c}.$$

Put differently, we would like to show that if the cut condition holds by a factor of c, then there is a valid MCF.

Recalling our definition of the min-ratio cut

$$\sigma(G) = \min_{\substack{U \subseteq V \\ |U| \leq \frac{n}{2}}} \frac{e(U, V-U)}{|U||V-U|},$$

we see that this is just a special case of the narrowest cut for the MCF problem where each pair of vertices is one of our (s_i, t_i) pairs. This is because the number of pairs split by a cut $(U, V - U)$ is just $|U||V - U|$.

Definition 3.3 *For an undirected graph G with unit capacity edges, let $\lambda^*(G)$ be the maximum amount of flow that can be sent simultaneously between all pairs of points.*

Claim 3.2 specialized to this case becomes

$$\lambda^*(G) \leq \sigma(G)$$

The following famous and very useful theorem is the approximate max flow min cut theorem we were looking for, in this special case.

Theorem 3.4 (Leighton-Rao, '88) $\lambda^*(G) \geq \frac{\sigma(G)}{O(\log n)}$*, and there is a polytime algorithm to find a cut of ratio $O(\lambda^*(G) \log n)$.*

The following example shows that this theorem is tight. Let G be from a family of expander graphs with maximum degree at most d and edge expansion $\alpha(\cdot)$ at least ϵ (where d and ϵ are constants). Then $\sigma(G) = \frac{\alpha(G)}{\Theta(n)} \geq \frac{\epsilon}{\Theta(n)}$. We want to show $\lambda^* \leq \frac{1}{\Theta(n \log n)}$, i.e. we cannot send more than $\frac{1}{\Theta(n \log n)}$ units of flow between all pairs.

Note that, because the maximum degree is d, a constant fraction of all pairs of nodes are separated by distance $\Omega(\log n)$. So for these $\Theta(n^2)$ pairs, each unit of flow consumes $\Omega(\log n)$ units of capacity. Since G has only at most $\frac{1}{2}dn$ edges, we have $\Theta(\lambda^* n^2 \log n) \leq \Theta(n)$, so $\lambda^* \leq \Theta(\frac{1}{n \log n})$.

4 LP formulation of max concurrent flow

While we could write a compact (i.e. polysize) LP formulation of maximum concurrent flow in terms of edge flows, from the theoretical view it is more useful to write an exponential size LP in terms of flows along paths. Let $\mathcal{P}_i$ denote the set of paths from s_i to t_i, and $\mathcal{P} = \bigcup_i \mathcal{P}_i$. We have a decision variable x_P for each $P \in \mathcal{P}$, and the LP is

$$\begin{array}{rrl} & \max \lambda & \\ \text{s.t.} & \sum_{P \ni e} x_P \leq 1 & \text{for all } e \in E \\ & \lambda - \sum_{P \in \mathcal{P}_i} \leq 0 & i = 1, 2, \ldots, k \\ & x, \lambda \geq 0. & \end{array}$$

The dual LP has a variable y_e for each edge, which we interpret as an edge length, and a variable z_i for each commodity, which we interpret as the length of the shortest path between

s_i and t_i, with respect to the lengths y_e. The dual LP is

$$\begin{aligned} & \min \textstyle\sum_e y_e \\ \text{s.t.} \quad & \textstyle\sum_{e \in P} y_e - z_i \geq 0 \quad \text{for all } P \in \mathcal{P}_i \\ & \textstyle\sum_i z_i \geq 1 \\ & y, z \geq 0. \end{aligned}$$

We can interpret the dual problem as trying to assign as little total edge length as possible such that

$$\sum_i \text{dist}_y(s_i, t_i) \geq 1.$$

Then sending λ units of flow between *all* (s_i, t_i) pairs burns up λ units of volume in the network (using the same analogy as in subsection 2.1), so $\lambda \leq \sum y_e$. LP strong duality gives $\max \lambda = \min \sum y_e$.

Intuitively, this says that to find good cuts, we should round up the long edges (i.e. include them in the cut), since those are the ones that represent the biggest bottlenecks to flow. It is tricky to transform this intuition into an effective algorithm, which is what we will address next lecture.