CS 388R: Randomized Algorithms Fall 2015

Lecture 22 — 23 November 2015

Prof. Eric Price *Scribe: Sid Kapur, Neil Vyas*

1 Overview

In the last lecture we studied graph sparsifiers. In this lecture we will study Network Coding and Edge Connectivity.

2 Edge Connectivity

Define the s-t edge-connectivity to be the number of disjoint paths from s to t. Note that this is the same as the size of the s-t min-cut, which we will denote as $C_{s,t}$. Today, we will study s-t edge-connectivity on DAGs, or Directed Acyclic Graphs.

Naively, we can use the Ford-Fulkerson algorithm, where we greedily choose paths at each step, flipping those paths that we used to move from s to t. This achieves $O(md)$ time, where m is the number of edges, and d is the max flow from s to t.

Suppose we are interested in the following two variants of this problem: all-pairs and single-source edge-connectivity. Then, using the algorithm described above, we can achieve $O(mdn^2)$ and $O(mdn)$ time, respectively, where n is the number of vertices and m and d are as above. (Note: we can replace d by any quantity that upper bounds the max flow, for example, the maximum vertex degree.)

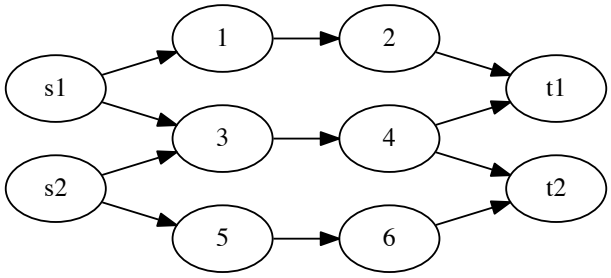
Can we improve? We claim that we can achieve single-source edge-connectivity in $O(md^2)$. (Due to Cheung, Lau, and Leung, 2011). In fact, we can achieve acyclic, single-source edge-connectivity in $O(md^{\omega-1})$ and all-pairs (not necessarily acyclic) in $O(m^\omega)$, where ω is the time complexity of matrix multiplication; we upper bound this by 3 to make the first claim.

The idea behind this analysis is to relate the edge-connectivity problem to network coding.

3 Network Coding

3.1 Introduction

In each time step, each node can send one "packet" of information, and each edge is capable of transmitting one "packet." Say that we have the following graph, and s_1, s_2 have messages A, B respectively, which they want to send to t_1 and t_2. Our objective will be to saturate the edges of the min-cut, since those are our "bottlenecks." We can be clever by sending A along the top path, B along the bottom path, and $A \oplus B$ along the middle path, and decoding B, A at t_1, t_2, respectively, using $A \oplus B$.



Suppose that we now want to send a message from s_1 to t_1, and another from s_2 to t_2. Then we can send $\frac{2}{3}$ of each source's message to the non-shared branch and $\frac{1}{3}$ to the shared branch, achieving $\frac{1}{2}\ell$ bits communicated per round.

In fact, we can send $\geq \ell C_{s,t}(T-n)$ bits in T rounds, where ℓ is the number of bits each node can communicate (size of the message, in our case), and n is the length of the path the messages travel.

3.2 Unknown / Changing network structure

The earlier methods relied on our knowledge of the network's structure to achieve more efficient communication. However, we might not know the structure of the network, or it might be changing in time. Thus, we would like a method that is oblivious to the overall network structure. We will employ the idea of sending components of the messages obliviously and then reconstructing the full message at the destination. Suppose we have many source nodes, with distinct messages, trying to communicate with one destination t.

Suppose that our input is of the form

$$m_1, \ldots, m_k \in \mathbb{F}_q^r;$$

note that this input is $kr \log q$ bits in size.

For $i = 1, \ldots, k$, let $\hat{m}_i = (e_i, m_i) \in \mathbb{F}_q^{r+k}$, where e_i is the unit vector in $\mathbb{F}_q^k$. Note that $\hat{m}_1, \ldots, \hat{m}_k$ are linearly independent because of the e_is. This guarantees that we can recover the original vectors exactly through Gram-Schmidt. Now, let $X_s = \text{span}\,\{\hat{m}_1, \ldots, \hat{m}_k\} \subseteq \mathbb{F}_q^{r+k}$.

Our goal is now to transmit X_s efficiently to t. This is a simpler problem because, in order to transmit the original messages, we only need to transmit k linearly independent vectors in X_s. This is sufficient to recover the basis $\hat{m}_1, \ldots, \hat{m}_k$, and from there our original messages.

Suppose that the capacity of each edge is $(k+r)\log q$ bits, i.e. we can transmit one vector in the subspace per edge per time step.

function SUBSPACECOMMUNICATION
  Set X_s as described above
  $X_v := \{0\} \quad \forall v \neq s$
  for all $(v, v') \in$ DAG-ordered edges **do**
    v samples $m \in X_v$
    v sends m to v'
    v' "adds" m to $X_{v'}$
  end for
  return X_t
end function

Claim 1. *There is a $1 - \frac{m}{q}$ chance that v' "learns" something from v in this procedure, given that $X_{v'} \neq X_v$.*

Proof. This is because there exists a u such that $u \in X_v, u \notin X_{v'}$, and in order for v' to "not learn" anything from v, the coefficient of u in m, the message that v sends to v', must be 0. Since we are in the field $\mathbb{F}_q$, this happens with probability $\frac{1}{q}$. □

Note that the communication cost is $kr \log q$ (which grows slowly with q), so we can set q to be a very large prime without a great penalty.

Claim 2. *Let C be the number of disjoint paths from s to t (by our notation above, there are $C_{s,t}$ such paths). Then for all vertices $v_1, \ldots, v_C$, where v_i is in the ith of the C disjoint paths, there exist vectors $u_1, \ldots, u_C$, where $u_j \in X_{v_j}$, and X_{v_j} is the subspace "by the time" v_j is reached, such that the u_i are linearly independent with probability $1 - \frac{m}{q}$.*

Proof. The proof follows by induction on time.

Suppose there exist $u_1, \ldots, u_C$ with $u_j \in X_{v_j}$, and the u_i are linearly independent. Send u from v_i to v_i'. Then we want to show that there exist $u_1 \in X_{v_1}, \ldots, u_C \in X_{v_C}$ such that these u_i are linear independent.

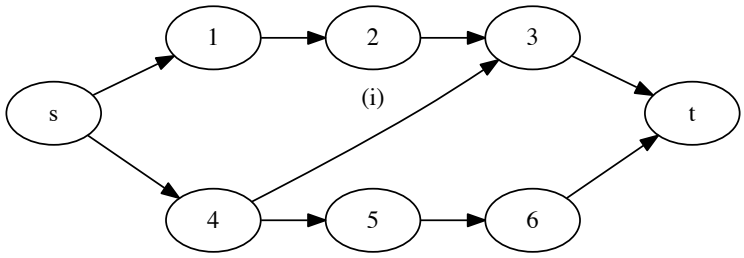
This is true if $u \perp \{u_1, \ldots, u_{i-1}, u_{i+1}, \ldots, u_C\}$, else this is true if $\langle u, u_i \rangle \neq 0$. Note that this second condition occurs with probability $1 - \frac{1}{q}$. Since we make at most m moves total, we have that "they all work," i.e. the vectors are linearly independent in each step, with probability $1 - \frac{m}{q}$. Note that m here is the length of the path from s to t, not the vectors of the input.

Thus, we need to make m steps, and in each step we perform $d^2 \log q$ work, since it takes $\log q$ time to operate on vectors in $\mathbb{F}_q$. So our time complexity for bit operations is $O(md^2 \log q)$ with success probability $1 - \frac{1}{m^c}$, and thus our time complexity for word operations is $O(md^2)$, as desired. □

Now we address the method of sampling u from X_v. Set $X_v = U_v^T Y$, where $Y \sim \mathbb{F}_q^{|X_v|}$ are i.i.d. uniform, and U_v are the vectors received by v up to this time, appropriately orthogonalized. Then let $u' = u - U_v U_{v'} u$. Note that the term being subtracted should remind you of projection. Then $u' = 0 \iff u \in X_{v'}$, so if $u' = 0$, set $U_{v'} = \left[\frac{U_v}{u}\right]$.

Note that we have, throughout, relied on some ordering of the vertices. This ordering is DAG ordering, in which we only transmit from a node if it has received all (possible) incoming messages.

For example, in the following graph, if we don't employ DAG ordering, it's possible that we reach the destination "too soon," and the subspace we've transmitted isn't of high enough dimension; suppose we take path (i), for example.



3.3 Continuous Transmission

This algorithm involves every vertex communicating at most once, so what about continuous transmission? We'll give a cursory look at the gossip algorithm.

After T rounds, t receives X with dimension $\geq C(T-n)$ w.p. $1 - \frac{(T-n)m}{q}$, where $k = C(T-n)$. This implies that we've sent $(r \log q)C(T-n)$ bits.

Claim 3. *$r \log(q)CT$ is optimal for any protocol.*