CS 388R: Randomized Algorithms, Fall 2019 November 14th, 2019

Lecture 23: Network Coding & Edge Connectivity

Prof. Eric Price *Scribe: Jeongyeol Kwon, Jiacheng Zhuo*

NOTE: THESE NOTES HAVE NOT BEEN EDITED OR CHECKED FOR CORRECTNESS

1 Overview

In the last lecture we studied a randomized algorithm to sparsify the dense graph using the notion of graph Laplacian and Rudelson-Vershynin inequality.

In this lecture we study the edge connectivity and network coding problems in a directed acyclic graph (DAG).

2 Problem Setup

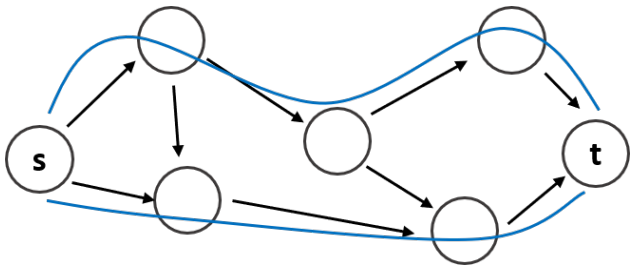

Figure 1: Directed Acyclic Graph (DAG) with 2 Edge Connectivities

2.1 Edge Connectivity

Consider a directed acyclic graph $G(E, V)$ with m edges and n vertices. The $s-t$ edge connectivity is defined as the maximum number of (edge) disjoint paths from s to t as in Figure 1. It can be computed with any algorithm (*e.g.,* Ford-Fulkerson) for finding max-flow/min-cut from s to t. Typical implementation of deterministic algorithms runs in $O(nm)$ computational complexity, where m can be as large as n^2. In this lecture, we study a randomized algorithm which gives $O(nd^{1.38})$ running time due to [CLL11], where d is the maximum degree of the graph.

2.2 Network Coding

The network coding problem considers finding routes in a directed graph G to send k messages $m_1, m_2, ..., m_k$ from s to t. Here, G may contain a cycle. At each time step, an edge $e(u, v)$ can

send one packet from u to v. We want to get all messages at the node t after R time steps, but we do not concern about the order in which messages arrive.

The easiest solution for the network coding problem is to use the maximum number of disjoint paths for different messages. Since the length of the longest path is at most n, we are guaranteed to send all messages after $R = n + k/C(s,t)$ steps, where $C(s,t)$ is the minimum cut from s to t. However, this deterministic solution suffers from two drawbacks: (1) it has to know the underlying structure of the graph in advance, (2) it is not robust to the potential packet loss that may occur due to any (physical) instability in the network.

3 Randomized Algorithm for Network Coding

Each message m_i consists of l letters where each letter is in q number system (that is, $m_i \in \mathbb{F}_q^l$), and each packet uses larger number of letters $l' = l + k > l$. We assume q and l are sufficiently large constants.

We first describe the algorithm, and then present the analysis. For the algorithm, we precede as follows:

1. Append headers to the messages:

$$\text{messages} = z = \left(\begin{bmatrix} 1 & 0 & 0 & m_1 \\ 0 & \ddots & 0 & \vdots \\ 0 & 0 & 1 & m_k \end{bmatrix} \right)$$

Denote the 'knowledge' of node v as X_v. For node s, $X_s = \text{span}(z)$. For every other node v, we initialize $X_v = \{0\}$.

2. At each time step for every edge (u,v), u picks random elements $w \in X_u$, send w over the edge, v update X_v to span $\left(\begin{bmatrix} X_v \\ w \end{bmatrix} \right)$.

Definition 1. ***(Awareness)*** *We say node v is aware of w if $\exists w' \in X_v$ such that $\langle w, w' \rangle \neq 0$.*

All we want is that, after the algorithm is terminated, the node t is aware of everything that s is aware of.

Lemma 2. *For any w, when u sends a message to v (u is aware of w), then the probability that v remain unaware of w afterwards is at most $\frac{1}{q}$.*

Now consider a single path of length L. After $L + r$ rounds,

$$\begin{aligned} &\mathbb{P}[v \text{ remains unaware of } w] \\ \leq &\mathbb{P}[\text{at least } r \text{ steps failed to make progress}] \\ \leq &\binom{L+r}{r} \cdot \frac{1}{q^r} \\ \leq &\left(\frac{e(1+L/r)}{q} \right)^r \end{aligned}$$

Then there are $C(s,t)$ dis-joint path.

$$\mathbb{P}[t \text{ is unaware of } w \text{ after } R = n + r \text{ rounds is}]$$
$$\leq \left(\frac{e(1+L/r)}{q}\right)^{rC(s,t)} \leq \left(\frac{2e}{q}\right)^{rC(s,t)} = \left(\frac{2e}{q}\right)^{(R-n)C(s,t)}$$

If $R \geq \frac{n}{\epsilon}$, and $q \geq 2^{1/\epsilon}$, we have

$$\mathbb{P}[t \text{ is unaware of } w \text{ after } R \text{ rounds is}] \leq q^{-RC(s,t)(1-O(\epsilon))}$$

We take a union bound on all the possible w:

$$\mathbb{P}[X_s = X_t] \geq 1 - q^k \cdot q^{-RC(s,t)(1-O(\epsilon))}.$$

We would like to have the probability be larger than $1 - 1/q$. Hence we need $k < RC(1 - O(\epsilon))$, which is an upper bound about how many packet we can send, given the graph structure.

How good is this upper bound? Consider how many packet we can send with R rounds of communication:

$$\#\text{packet} \leq R \cdot \frac{\ell'}{l} C(s,t) = R \cdot C(s,t)(1+\epsilon) \triangleq \text{OPT}$$

That is, the upper bound is only $(1-\epsilon)$ times of the OPT.

4 Randomized Algorithm for Edge Connectivity

We can apply the network coding algorithm to compute an edge connectivity in a DAG with maximum degree d. Since now we consider a DAG, we can perform a topological sort to get a sorted list of vertices. Then we will simulate a single round of the propagation of "awareness" in topological order. Assume we send d orthogonal messages (e.g. standard basis) of length d (hence $k = l = l' = d$). After one round of the propagation, we will show that $dim(X_t) = C(s,t)$ with high probability (note that $dim(X_t)$ can be at most $C(s,t)$).

Similarly to the analysis of the algorithm for Network Coding problem, consider the propagation of the awareness of any fixed $w \in \mathbb{F}_q^d$. Consider the maximum number of edge-disjoint paths from s to t. From this, the probability of w being transmitted from s to t is at least

$$1 - (1 - (1-1/q)^n)^{C(s,t)} \geq 1 - (n/q)^{C(s,t)} = 1 - q^{-C(s,t)(1-\log_q n)},$$

since for each path, the success probability is at least $(1-1/q)^n$.

Let N_w be the number of $w \in \mathbb{F}_q^d$ that node t is unaware of. Note that $N_w = q^{d-dim(X_t)}$. On the other side, the expected value of N_w is at most $q^{d-C(s,t)(1-\log_q n)}$. With Markov's inequality, we have that

$$P(N_w \geq q^{1/2} q^{d-C(s,t)(1-\log_q n)}) \leq q^{-1/2}.$$

That is, if $d - C(s,t)(1-\log_q n) + 1/2 < d - C(s,t) + 1$, then it implies $N_w = q^{d-dim(X_t)} < q^{d-C(s,t)+1}$, hence we have $dim(X_t) = C(s,t)$. After rearranging the relation, if

$$1/2 > C(s,t) \log_q n,$$

then we are done. Finally, we set $q > n^{2d}$ so that $C(s,t)\log_q n < d(1/2d) = 1/2$ (here we assume that we can do the operations with this large q in $O(1)$).

Now we consider the computational cost of this algorithm. At each edge from u to v, we sample a random vector in X_u, which takes $O(d^2)$ time to generate random coefficients and compute the vector. As we perform this operation m times, we achieve $O(md^2)$ running time.

In order to get $O(md^{1.38})$ computational complexity, we batch the coefficients that will be used at each node. That is, we prepare a $\mathbb{F}_q^{d\times d}$ coefficient matrix, and perform a matrix multiplication when we start a propagation at each node. Once this matrix multiplication is computed in $O(d^{2.38})$, we can use the result for all outgoing edges, hence we can save $O(d)$ factor and achieve $O(md^{1.38})$ running time.

References

[CLL11] Ho Yee Cheung, Lap Chi Lau, and Kai Man Leung, "Graph Connectivities, Network Coding, and Expander Graphs" *IEEE 52nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 190-199, 2011.