CS 388R: Randomized Algorithms, Fall 2023 November 13, 2023

Lecture 23: Network Coding

Prof. Eric Price *Scribe: Ruichen Jiang, James Rayman*

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

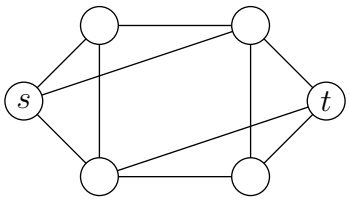
1 Overview

In this lecture we will discuss network coding, a method of transmitting information through a distributed system in a near-optimal way. Network coding is robust in two important ways: First, every node only needs to know the nodes it is connected to (as opposed to the entire network), and the algorithm will still perform near-optimally if the network changes midway through.

Our method of analysis is given by Bernhard Haeupler in [H16]. Today we will only analyze the case of a static network, but the method of analysis can easily be generalized for the dynamic case (as we see in HW 10).

2 Introduction

First, consider a graph of n nodes with two distinguished nodes s and t.



Define $C_{s,t}$ to be the s-t edge connectivity. Equivalently, $C_{s,t}$ is the unweighted s-t min cut, or the maximum number of edge-disjoint paths between s and t. For the graph above, $C_{s,t} = 3$. We will also write C to mean $C_{s,t}$ later in this lecture.

Now, let's solve a practical problem. Imagine that the graph is a distributed system and s wants to send a message of k packets to t. The network will be synchronized, and at every time step, each edge can send 1 packet in each direction. How do we send the message in as little time as possible? We will call the optimal time OPT.

For $k = 1$, the answer is the shortest path, but today, we will focus on cases where k is very large. In such a case, our throughput will be limited by the s-t min cut, since at every time step, at most C packets can travel across the cut. Thus, we have:

$$\text{OPT} \geq \frac{k}{C}$$

However, we can also find C edge-disjoint paths from s to t and evenly divide the packets to send along those routes. This means s sends out C packets per time step, and each of these packets will reach t within n steps of being sent. Thus, we also have an upper bound:

$$\text{OPT} \leq \frac{k}{C} + n$$

Since n is small compared to k, this bound is essentially tight. The algorithm of evenly dividing the packets along edge-disjoint paths is near-optimal, but it is fragile in two ways. First, many nodes will need some global information about the network in order to determine which ways to send packets, and second, if the network changes while packets are being transmitted, then the algorithm might not be near-optimal anymore.

Both of these issues can be resolved with an alternative randomized algorithm: network coding.

3 Network Coding

We first formally describe the problem. We assume that there are k **messages** $\vec{m}_1, \vec{m}_2, \ldots, \vec{m}_k$. Each message has ℓ characters over the finite field $\mathbb{F}_q$ (i.e, m_i is a vector from $\mathbb{F}_q^\ell$ of length ℓ for $i = 1, 2, \ldots, k$). Moreover, each **packet** is a vector from $\mathbb{F}_q^{\ell'}$, where $\ell' = \ell + k$. The goal is to send all the messages from s to t as quickly as possible, so that the total number of rounds R is at most $(1 + \epsilon) \cdot \text{OPT}$.

To give us an idea of the best possible result, we first establish a lower bound on OPT. The idea here is essentially the same as the one we have seen in the introduction, but we need to be a bit more careful since the message and the packet have different numbers of characters. Specifically, note that there are $k \cdot \ell$ characters in total that need to be sent from s and t. However, in each round, we can send at most one packet per edge that defines the minimum s-t cut. This means that at most $C \cdot \ell'$ characters can "flow" through the min-cut in each round. Therefore, we must have

$$\text{OPT} \geq \frac{k \cdot \ell}{C \cdot \ell'}. \tag{1}$$

Next, we describe the network coding algorithm. Every packet follows the format of $(\vec{\mu}, \vec{m}) \in \mathbb{F}_q^{\ell'}$, where $\vec{m} = \sum_{i=1}^{k} \mu_i \vec{m}_i \in \mathbb{F}_q^\ell$ is a linear combination of the k messages and $\vec{\mu} = (\mu_1, \ldots, \mu_k) \in \mathbb{F}_q^k$ is the vector of the coefficients[1]. Moreover, during the execution of the algorithm, every node v will maintain a subspace X_v in $\mathbb{F}_q^{\ell'}$, which is given by the linear span of all the packets it knows so far. Given these notations, the algorithm is given as follows.

- Initially, node s knows all the messages, while all the other nodes know nothing. Thus, we have

$$X_s = \text{span}\left(\left[\begin{array}{cccc:c} 1 & 0 & \cdots & 0 & \vec{m}_1 \\ 0 & 1 & \cdots & 0 & \vec{m}_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & \vec{m}_k \end{array}\right]\right) \subset \mathbb{F}_q^{\ell'}, \tag{2}$$

  and $X_v = 0$ for all other nodes v.

[1] In the following, we view $\vec{m}$ as a row vector.

- Then, in every round:
  - For every edge (v, w) in the graph, node v picks a random vector $\vec{u} \in X_v$ and send the packet $\vec{u}$ to node w.
  - At the end of the round, if node v receives the packets $\vec{u}_1, \vec{u}_2, \ldots, \vec{u}_p$ from its neighbors, it then updates the subspace by $X_v \leftarrow \text{span}(X_v, \vec{u}_1, \vec{u}_2, \ldots, \vec{u}_p)$.

A few remarks are in order. First, the destination node t will be able to decode all the messages $\vec{m}_1, \ldots, \vec{m}_k$ when it gets to know a subspace of dimension k. Indeed, it can first find a basis of X_t, use the Gram-Schmidt orthogonalization to obtain a basis matrix in the form of (2), and then read off the messages from the right block of the partitioned matrix. Thus, when t knows a k-dimensional subspace, it can send an acknowledgment signal to the rest of the network to terminate the algorithm.

Second, as promised, each node only needs to know its immediate neighbors to carry out the procedure above and does not require knowledge of the global topology of the graph. We can see that randomization plays a crucial role here: since the node v has no idea where each of its neighbors leads, it needs to deal with such uncertainty by sending random packets to its neighbors. Otherwise, say if the source node s always sends $\vec{m}_1$ to its first neighbor v_1, then this message could be lost in the network if there is no path from v_1 to t.

Finally, the procedure works even if the network is dynamic (under certain constraints), but we will not prove this today.

4 Analysis

In the section, we will show that the network coding algorithm in Section 3 matches the optimal time in (1) up to a multiplicative error. The trick is to analyze what is not in the subspace. For each vertex v, we define the orthogonal subspace as

$$Y_v = X_v^{\perp} = \{\, \vec{u} \in \mathbb{F}_q^{\ell'} : \langle \vec{u}, \vec{u}' \rangle = 0 \quad \forall \vec{u}' \in X_v \}.$$

Definition 1 ("Awareness")**.** *We say the node v is "aware" of $\vec{u}$ if we have $\vec{u} \notin Y_v$.*

Lemma 2. *Assume that node v is "aware" of $\vec{u}$, node w is "unaware" of $\vec{u}$, and v sends a packet to w. Then*

$$\mathbb{P}[w \text{ remains unaware of } \vec{u}] \leq \frac{1}{q}.$$

Proof. Let $\vec{u}'$ be the packet sent. w is only unaware of $\vec{u}$ after receiving $\vec{u}'$ if $\vec{u}$ is orthogonal to $\vec{u}'$. That is, if $\langle \vec{u}, \vec{u}' \rangle = 0$.

To calculate this probability, first find a basis of X_v, let's call it $(\vec{a}_1, \ldots \vec{a}_d)$. Then $\vec{u}'$ is of the following form:

$$\vec{u}' = z_1 \vec{a}_1 + z_2 \vec{a}_2 + \cdots + z_d \vec{a}_d$$

where each of the $z_i \in \mathbb{F}_q$ are independent uniform random variables. Since v is aware of $\vec{u}$, at least one of the basis vectors is not orthogonal to $\vec{u}$. WLOG suppose $\vec{a}_1$ is such a basis vector.

Let's fix all the z_i except z_1. We have:

$$\begin{aligned}\langle \vec{u}, \vec{u}' \rangle &= \langle \vec{u}, z_1 \vec{a}_1 + z_2 \vec{a}_2 + \cdots + z_d \vec{a}_d \rangle \\ &= z_1 \langle \vec{u}, \vec{a}_1 \rangle + \langle \vec{u}, z_2 \vec{a}_2 + \cdots + z_d \vec{a}_d \rangle \end{aligned} \tag{3}$$

Since $\vec{u}$ and $\vec{a}_1$ are not orthogonal, $\langle \vec{u}, \vec{a}_1 \rangle \neq 0$. So, since we are working in a field, there is exactly one value of z_1 such that (3) is not zero. Thus, there is at most a $1/q$ chance that w remains unaware of $\vec{u}$. □

As a consequence of this, awareness spreads like wildfire! To begin with, consider a particular path from s to t consisting of L edges. Then after $L + r$ rounds:

$$\begin{aligned}\mathbb{P}[t \text{ remains unaware of } \vec{u} \text{ by this path}] &\leq \mathbb{P}[\text{at least } r \text{ of the } L + r \text{ steps fail}] \\ &\leq \binom{L+r}{r} \cdot \frac{1}{q^r} \\ &\leq \left(\frac{e(1 + L/r)}{q} \right)^r \approx q^{-r}. \end{aligned} \tag{4}$$

Thus, as r increases, t will become aware of $\vec{u}$ with high probability.

Now recall that there are C edge-disjoint paths in total between s and t, and the packets sent along each path are independent from each other. Since each path has length at most n, after $R = n + r$ rounds, we have

$$\mathbb{P}[t \text{ remains unaware of } \vec{u}] \leq \left(\frac{e(1 + n/r)}{q} \right)^{Cr}. \tag{5}$$

To simplify the expression, assume that we are in the regime of "big messages" and "lots of time", that is,

$$\log q \gtrsim \frac{1}{\epsilon}, \quad \ell \gtrsim \frac{k}{\epsilon}, \quad \text{OPT} \gtrsim \frac{n}{\epsilon}.$$

Moreover, we choose $r \geq n$ such that $R = n + r \geq n/\epsilon$ (we will give the specific choice of R later). Then we can further bound the probability in (5) by

$$\begin{aligned}\mathbb{P}[t \text{ remains unaware of } \vec{u}] &\leq \left(\frac{2e}{q} \right)^{C(R-n)} && (\text{since } r \geq n \text{ and } r = R - n) \\ &\leq \left(\frac{2e}{q} \right)^{CR(1-\epsilon)} && (\text{since } R \geq n/\epsilon) \\ &\leq \left(\frac{1}{q} \right)^{CR(1-\epsilon)^2} && (\text{since } \log q \gtrsim \frac{1}{\epsilon}) \end{aligned}$$

In the last inequality, we used the fact that $\log q \gtrsim \frac{1}{\epsilon} \Rightarrow q^\epsilon \gtrsim 1 \Rightarrow 2e \leq q^\epsilon$.

So far, we have bounded the probability that t is unaware of a fixed vector $\vec{u} \in X_s$. Moreover, for X_t to reach dimension k, it is sufficient to show that t becomes aware of all the vectors in X_s. To see this, suppose the dimension of X_t is less than k. Since X_t is a subspace of X_s, this means that there exists a vector $\vec{u} \in X_s$ that is orthogonal to X_t, which, by definition, means t is unaware of $\vec{u}$. This leads to a contradiction.

Since X_s is a k-dimensional subspace over the finite field $\mathbb{F}_q$, there are q^k distinct vectors in X_s. Thus, we further have

$$\mathbb{E}[\#u \in X_s \text{ that } t \text{ is unaware of}] \leq q^k \cdot (q)^{-CR(1-\epsilon)^2} = q^{k-CR(1-\epsilon)^2}.$$

Therefore, if we set $R > \frac{k+c}{C(1-\epsilon)^2} = \frac{k}{C}(1+\mathcal{O}(\epsilon))$ for a constant c, the expected number of vectors in X_s that t is unaware of is less than q^{-c}, which implies that t is aware of all the vectors in X_t with probability at least $1-q^{-c}$, i.e. with high probability. On the other hand, since $\ell \gtrsim \frac{k}{\epsilon}$, we get from the bound in (1) that $\text{OPT} \geq \frac{k}{C}(1-\epsilon)$. Putting all together, we obtain $R = \text{OPT} \cdot (1+\mathcal{O}(\epsilon))$, which also satisfies the requirement that $R \geq n/\epsilon$. This completes the proof.

5 Closing Remarks

We have analyzed network coding only in the static case today, but we can easily generalize our analysis since the only part that really changes is the step used to derive (4).

As a final note, under certain constraints, simulating network coding on a static network is a good way to estimate s-t edge connectivity on a given graph.

References

[H16] Bernhard Haeupler. Analyzing Network Coding (Gossip) Made Easy. *J. ACM*, 63(3):1–22, 2016.