CS 388R: Randomized Algorithms, Fall 2021 November 11, 2021

Lecture 22: Network Coding

Prof. Eric Price *Scribe: Yeongwoo Hwang and Geoffrey Mon*

1 Overview

In this lecture we discuss *Network Coding*, which solves the problem of transmitting a message composed of multiple *words* from a source vertex s to a target vertex t in some graph G. In fact, we'll show that the full source message will, with high probability, be received by every vertex in the graph.

2 Algorithms for Network Coding

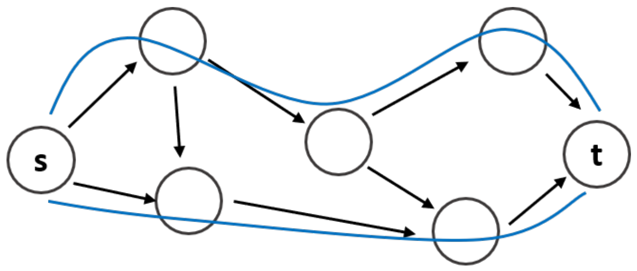

Figure 1: Directed Acyclic Graph (DAG) with 2 Edge Connectivities

First, we'll formally define the problem. The source vertex s "knows" messages $m_1, \ldots, m_k \in \mathbb{F}_q^l$, where q is the size of a single word (e.g. $q = 2^64$ for 64-bit words) and l is the number of words per "packet." k is the total number of packets in the file that s is sending. At any given round, an edge $e \in E(G)$ can send a message $v \in \mathbb{F}_q^{l'}$, where $l' > l$. One can think of v as including some message m_i as well as a $(l' - l)$-word *header*. We are then interested in computing the number of messages m_i t can receive as a function of R, the total number of rounds.

2.1 Naive Algorithm

A very simple algorithm proceeds as follows. Each $v \in V(G)$ keeps track of all the messages its received so far; call this set $X_v = \{(i, m_i) : i \text{ received by } v\}$. At the beginning of the algorithm, $X_s = \{(1, m_1), \ldots, (k, m_k)\}$ and for $v \neq s$, $X_v = \emptyset$. At any given round, each vertex sends to all of its neighbors the *highest* indexed message its received thus far. In this simple algorithm, at round R, t receives

$$R - \text{dist}(s, t) \geq R - n$$

messages. This is because there's a lag of $\text{dist}(s,t)$ for the first message to reach t from s. This seems, pretty good, an in particular, as $R \to \infty$, t receives $(1-o(1))R$ messages. But we can do better!

2.2 Better (Non-Randomized) Algorithm

The intuition we'll use is for our improved, non-randomized (and later, randomized) algorithm is that we can send different messages along edge-disjoint paths. In the example graph above, we could, for instance, send even-indexed messages along the top path and odd-indexed messages along the bottom path. Letting $C_{s,t}$ be the minimum s-t cut, by max-flow/min-cut duality, there are also $C_{s,t}$ edge-disjoint paths from s to t. Thus, we get an improved rate of

$$C_{s,t} \cdot R \cdot (1-o(1))$$

Unfortunately, this algorithm has the downside of needing to "pre-process" the graph and exactly map out the disjoint paths. This is an expensive operation, and if the graph changes we might need to do it many times. Network coding solves this issue.

2.3 Randomized Algorithm

Network coding is a simple, robust algorithm which is *oblivious* to the underlying graph. In particular, each vertex u only cares about its immediate neighbors $N(v)$. We'll show that this algorithm also obtains $C_{s,t} \cdot R \cdot (1-o(1))$ for sending the desired messages to any given destination vertex. This is actually the best possible result. In a single round, we can transmit at most l' messages across each edge of the size-$C_{s,t}$ cut. Over R rounds, this gives an upper bound of $C_{s,t}Rl'$ words sent. Sending k l-word messages requires kl words, so

$$kl \le C_{s,t}Rl' \implies k \le C_{s,t}\frac{Rl'}{l}$$

For large enough l, this essentially matches the lower bound given by Network Coding. Now we give the actual algorithm.

We'll set $l' = l + k$, with the header being the *unary* representation of the message index. That is, we concatenate a unary representation to each m_i, such that the total spaces of "known" l'-word messages by s is then,

$$\text{span}\left(\left[\begin{array}{ccccc|c} 1 & 0 & 0 & \dots & 0 & m_1 \\ 0 & 1 & 0 & \dots & 0 & m_2 \\ \vdots & & & & & \vdots \\ 0 & 0 & 0 & \dots & 1 & m_k \end{array}\right]\right) \subseteq \mathbb{F}_q^{l'} \tag{1}$$

Furthermore, we'll let each vertex v "know" a subspace $X_v \subseteq \mathbb{F}_q^{l'}$ at each point in time. X_v denotes the subspace spanned by messages that v has received. Initially, $X_v = \emptyset$. We will maintain that $X_v \subseteq X_s$. At any round, each vertex v selects a random $u \in X_v$ (or nothing if X_v) is empty, and sends it to its neighbors. At the conclusion of the round, each vertex v has received messages $(u_1, \dots, u_{d(v)})$ and updates its subspace as follows:

$$X_v = \text{span}(X_v \cup \{u_1, \dots, u_{d(v)}\})$$

That is, v updates its subspace by considering the span of some basis for X_v combined with all of the messages it has seen. Note that the matrix representing the messages v has received will almost certainly not be in the form given by equation (1). To put its matrix into the canonical form, v can diagonalize some basis for X_v using Gram-Schmidt orthogonalization; if $X_v = X_s$, then v can recover all of the k original messages $m_1, \dots, m_k$ by reading off of the right side of the matrix.

2.3.1 Analysis

To analyze this algorithm, we will need to prove that a given vertex v will grow its own subspace X_v quickly, until $X_v = X_s$. Let $Y_v = X_v^\perp = \{u \in \mathbb{F}_q^{l'} : \forall u' \in X_v.\ u \cdot u' = 0\}$, that is, the set of vectors that are orthogonal to everything in X_v. Say that v is "aware" of vector u if $u \notin Y_v$. Note the following:

- The number of vectors that v knows is $q^{\dim(X_v)}$, because we can consider all linear combinations of a basis of size $\dim(X_v)$
- The number of vectors that v is unaware of is $q^{l' - \dim(X_v)}$, because there is a basis of size $l' - \dim(X_v)$ for $\mathbb{F}_q^{l'} \setminus X_v$
- The number of vectors that v is aware of is $q^{l'} - q^{l' - \dim(X_v)} = q^{l'} \cdot (1 - 1/q^{\dim(X_v)})$ which we compute by subtracting the number of unaware vectors from the total size of $\mathbb{F}_q^{l'}$

Lemma 1. *If v is aware of u, w is unaware of u, and v sends a (random) message to w, then w becomes aware of u with probability $\geq 1 - 1/q$.*

Proof. First, note that the probability that w becomes aware of u is at least the probability that v sends u' such that $u \cdot u' \neq 0$, since when u' is added to X_w, u will no longer be orthogonal to everything in X_w. Then, for any u such that v is aware of u, a random u' has that $u \cdot u' = 0$ with probability $1/q$. Intuitively, you can imagine considering some basis that includes u; the probability that u' is orthogonal to u is the probability that we select 0 as the coefficient for u when picking a random linear combination. So, the probability that a random u' has $u \cdot u' \neq 0$ is $1 - 1/q$, which completes the proof. $\square$

Now, consider some path of length L from s to t. After $L + r - 1$ rounds, for an arbitrary fixed vector u,

$$\mathbb{P}[t \text{ is unaware of } u \text{ via this path}] \leq \mathbb{P}[u \text{ has failed to proceed } r \text{ times}]$$

That is, for r vertices on the path, that vertex (which was aware of u) failed to make the next vertex on the path aware of u. We can upper bound this by considering a specific combination of r rounds (using the fact that r independent rounds all fail to proceed with probability $\leq q^{-r}$), and then union bounding over all combinations.

$$\begin{aligned} &\leq \binom{L+r}{r} \cdot \frac{1}{q^r} \\ &\leq \left(\frac{e \cdot (1 + L/r)}{q}\right)^r \\ &\leq \left(\frac{2e}{q}\right)^r \end{aligned}$$

where the last inequality follows if we assume $r \geq L$.

Now, suppose we have $R = n + r$ total rounds. Set r such that $R \geq n/\epsilon$ for some ϵ which will appear later (and which we can set at the end). Then, we can bound the probability that t is unaware of some fixed vector u after R rounds. Note that there are $C_{s,t}$ edge-disjoint paths from s to t, and failures to proceed along these paths are independent between paths. So, t is unaware of u if all of these $s \to t$ paths have failed to proceed enough times:

$$\mathbb{P}[t \text{ unaware}] \leq \left(\frac{2e}{q}\right)^{rC_{s,t}} \leq \left(\frac{2e}{q}\right)^{C_{s,t}R(1-\epsilon)} \leq q^{-C_{s,t}R(1-\epsilon)^2}$$

where the last inequality follows from setting $q \geq 2^{O(1/\epsilon)}$, since we have that $q^\epsilon \geq 2^{O(1)} \geq 2e$, from which it follows that $2e/q \leq q^{-(1-\epsilon)}$.

Finally, X_s has q^k vectors because the dimension of X_s is k, and there are q choices for the coefficient for each of the k basis elements when assembling a linear combination. So, the expected number of vectors in X_s that t is unaware of after R rounds is $\leq q^{k-C_{s,t}R(1-\epsilon)^2}$, which is $\ll 1$ when $k < C_{s,t}R(1-\epsilon)^3$. So we have that with high probability, t is aware of all vectors in X_s: $\forall u \in X_s . \exists u' \in X_t . u \cdot u' \neq 0$. Because $X_t \subseteq X_s$, this implies $X_t = X_s$ i.e., t knows all of the original messages $m_1, \ldots, m_k$.

Recall that we can transmit at most l' messages across each edge of the $C_{s,t}$-sized cut. So, over R rounds, we must have that $k \leq C_{s,t} \cdot \frac{Rl'}{l} = C_{s,t}R \cdot \frac{l+k}{l}$. If l is large enough, we can say $l+k \leq (1+\epsilon)l$. Then, we have that $k \leq C_{s,t}R(1+\epsilon)$, which implies for a fixed k that we need $R \geq \frac{k}{C_{s,t}(1+\epsilon)}$. How does that compare to our randomized algorithm? We require that $R \geq n/\epsilon$, $q \geq 2^{O(1/\epsilon)}$, and $l \geq k/\epsilon$; putting these together, we have that $R = \frac{k}{C_{s,t}} \cdot (1 - O(\epsilon))$. So, we are within a $(1 + O(\epsilon))$ factor of the optimal number of rounds needed to send our k messages.

As an aside, note that this algorithm is robust to unreliable or changing networks. For example, if each transmission drops out with probability 1/2, that would only change where we analyze failures to proceed. Problem Set 9 contains a problem that explores how to analyze the performance of this algorithm on a dynamic network.