Coding in Undirected Graphs Is Either Very Helpful or Not Helpful at All

Mark Braverman[*1], **Sumegha Garg**[2], **and Ariel Schvartzman**[3]

1 **Princeton University, Princeton, USA**
`mbraverm@princeton.edu`
2 **Princeton University, Princeton, USA**
`sumeghag@cs.princeton.edu`
3 **Princeton University, Princeton, USA**
`acohenca@cs.princeton.edu`

Abstract

While it is known that using network coding can significantly improve the throughput of directed networks, it is a notorious open problem whether coding yields *any* advantage over the multicommodity flow (MCF) rate in undirected networks. It was conjectured in [11] that the answer is 'no'. In this paper we show that even a small advantage over MCF can be amplified to yield a near-maximum possible gap.

We prove that any undirected network with k source-sink pairs that exhibits a $(1+\varepsilon)$ gap between its MCF rate and its network coding rate can be used to construct a family of graphs G' whose gap is $\log(|G'|)^c$ for some constant $c < 1$. The resulting gap is close to the best currently known upper bound, $\log(|G'|)$, which follows from the connection between MCF and sparsest cuts.

Our construction relies on a gap-amplifying graph tensor product that, given two graphs G_1, G_2 with small gaps, creates another graph G with a gap that is equal to the product of the previous two, at the cost of increasing the size of the graph. We iterate this process to obtain a gap of $\log(|G'|)^c$ from any initial gap.

1998 ACM Subject Classification G.2.2. Graph Theory, Network problems

Keywords and phrases Network coding, Gap Amplification, Multicommodity flows

Digital Object Identifier 10.4230/LIPIcs.ITCS.2017.18

1 Introduction

The area of network coding addresses the following basic problem: in a distributed communication scenario, can one use coding to outperform packet routing-based solutions? While the problem of communicating information over a network can be viewed as the process of moving information packets between terminals, a key distinction between moving packets and moving commodities is that information packets can be re-encoded by intermediate nodes. For example, a node which receives packets P_1 and P_2 can calculate and transmit the bitwise XOR packet $P_1 \oplus P_2$ to its neighbor. This operation has no analogue in multicommodity flow scenarios.

[*] Research supported in part by an NSF Awards, DMS-1128155, CCF-1525342, and CCF-1149888, a Packard Fellowship in Science and Engineering, and the Simons Collaboration on Algorithms and Geometry. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

Whether (and to what extent) this ability confers any benefits over the simple routing-based solution, depends on the specific goal of the communication at hand. Such goals may include uni-cast and multi-cast throughput, error-resilience and security, to name a few. These questions have been the subject of active study in the recent past. A summary of major directions can be found in the books [14, 13], and surveys [5, 15].

In this paper we focus on noiseless unicast communication. The network is a capacitated graph G with k source-sink terminal pairs (s_i, t_i). Each source vertex s_i wants to transmit an information stream to t_i. The network coding rate $\mathrm{NC}(G)$ is the maximum rate at which transmission between all pairs can happen simultaneously, given the capacity constraints [2].

If we forbid coding, and restrict nodes to forwarding information packets that they receive, the problem becomes equivalent to multicommodity flow over G — the very well-studied problem of maximizing the rate $\mathrm{MCF}(G)$ at which commodities are moved from sources to sinks subject to the capacity constraints (see e.g. [3] for background). Clearly, the multicommodity rate can always be achieved — but can it be beaten using "bit tricks"?

If the graph G is directed, there are well-known examples which show that coding can improve throughput in a very dramatic way [7, 1]. There is a family of examples G, where the gap between the multicommodity flow rate and network coding throughput is as large as $O(|G|)$. Despite substantial effort, it is not clear whether coding confers *any* benefit over routing in undirected networks. Li and Li [11] conjectured that the answer is 'no'. This conjecture is currently open.

It is known that the Li and Li conjecture holds in some special cases. Naturally, it holds whenever the sparsity of the graph matches the multicommodity flow rate. For cases where these quantities are not equal, [8] and [9] show that the conjecture is true for the Okamura-Seymour graph and [8, 1] show it for an infinite family of bipartite graphs. Empirical evidence also suggests that the conjecture is true [12].

One simple case where coding rate cannot exceed capacity is the case when the channel is a single edge: two parties cannot be sending messages to each other at a total rate exceeding the channel's capacity. This is a simple consequence of Shannon's Noiseless Coding Theorem. As a simple corollary, the sparsest cut in G provides an upper bound for the network coding rate $\mathrm{NC}(G)$. The sparsity of a cut $(U, V \setminus U)$ is defined as

$$\mathrm{Sparsity}(U, V \setminus U) := \frac{\mathrm{Capacity}(U, V \setminus U)}{\mathrm{Demand}(U, V \setminus U)} \tag{1}$$

If we merge the vertices on either side of the cut, the network coding rate becomes $\mathrm{Sparsity}(U, V \setminus U)$. Merging nodes can only increase network coding rate, and thus we have $\mathrm{NC}(G) \leq \mathrm{Sparsity}(U, V \setminus U)$. Since the sparsity of G is defined as the minimum of (1) over all cuts, we have $\mathrm{NC}(G) \leq \mathrm{Sparsity}(G)$.

As discussed below, the multicommodity flow problem is very well-studied. In the one commodity case, the Max-Flow Min-Cut Theorem asserts that sparsity is equal to the flow rate. In the multicommodity case, the sparsity is still an upper bound on the multicommodity flow rate $\mathrm{MCF}(G)$, but it might be loose by a factor of $\log |G|$:

$$\mathrm{Sparsity}(G)/O(\log |G|) \leq \mathrm{MCF}(G) \leq \mathrm{Sparsity}(G). \tag{2}$$

Thus the advantage one can gain for network coding over undirected graphs is at most $O(\log |G|)$:

$$\mathrm{NC}(G)/O(\log |G|) \leq \mathrm{Sparsity}(G)/O(\log |G|) \leq \mathrm{MCF}(G) \leq \mathrm{NC}(G). \tag{3}$$

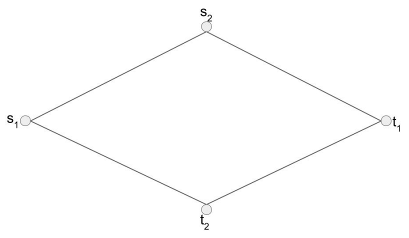


Figure 1 Graph G_1 and G_2.

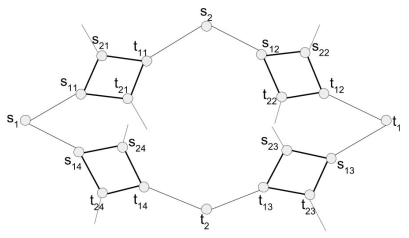


Figure 2 Basic gadget that embeds a copy of G_2 into each edge of G_1. The edges coming out of the copies will be used to connect to other copies of the outer graph (G_1). Labelled source-sink pairs of G_2 are just for reference. The thin edges are just an artifact and their respective end points represent a single vertex.

The Li and Li conjecture asserts that the rightmost '$\leq$' is indeed an equality. Our main result is that either the conjecture is true, or it must be nearly 'completely false': the gap between $\mathrm{NC}(G)$ and $\mathrm{MCF}(G)$ can be as high as poly-logarithmic in $|G|$.

▶ **Theorem 1.** *Given a graph G that achieves a gap of $1+\epsilon$ between the multicommodity flow rate and the network coding rate, we can construct an infinite family of graphs $\widetilde{G}$ that achieve a gap of $O\left(\log|\widetilde{G}|\right)^c$ for some constant $c<1$ that depends on the original graph G.*

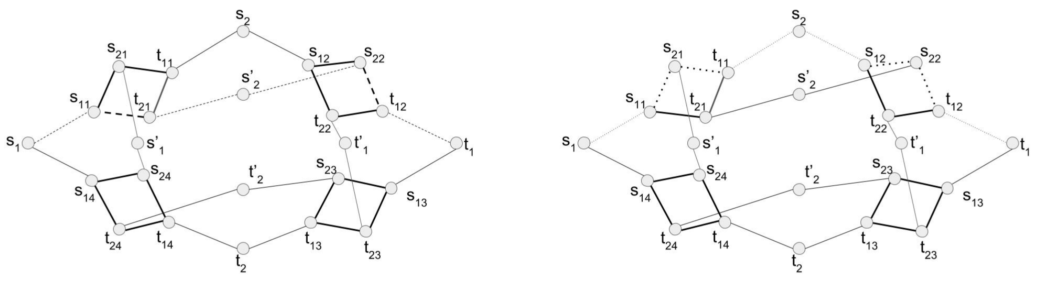
In order to prove Theorem 1, we will show a simpler construction that can be applied repeatedly.

▶ **Theorem 2.** *Given a graph G of size n with a gap of $1+\epsilon$ between the multicommodity flow rate and the network coding rate, we can create another graph G' of size n^{c^2} and a gap of $(1+\epsilon)^2$, where c depends on the diameter of the graph G.*

The idea of gap-amplification in the context of network coding has been studied before. In [4] the authors combine linear programming techniques with hypergraph product operations to prove general gap amplification results between linear and non-linear network coding for directed graphs.

Proof outline

The main part of the construction is to define a *graph tensor* on graphs G_1 and G_2 that have gaps of $1+\epsilon_1$ and $1+\epsilon_2$ respectively, between the multicommodity flow rate and the network coding rate, which gives a new graph G with a gap of $(1+\epsilon_1)(1+\epsilon_2)$ while keeping a check on the size of G. We can then take a graph with a small gap and tensor it with itself to produce a graph with an even larger gap. Repeatedly tensoring the output of the previous iteration with itself will give us Theorem 1.



(a) A cheating path is highlighted with dashed edges.

(b) An honest path is highlighted with dotted edges.

Figure 3 Two copies of G_1 (with source-sink pairs $s_1 - t_1, s_2 - t_2$ and $s'_1 - t'_1, s'_2 - t'_2$, respectively) have 8 edges. These edges are replaced with 4 copies of G_2, each copy having two source-sink pairs. All edges in the first copy of G_1 are replaced with s_{1X}-t_{1X} source-sink pairs of G_2; edges in the second copy of G_2 are replaced with s_{2X}-t_{2X} source-sink pairs of G_2

In G_2, network coding allows us to send more information from every source to its corresponding sink than what simple flows allow. We construct a gadget for the graph tensor exploiting this fact as in Figure 2. We replace each edge of G_1 by a copy of G_2 with endpoints at a deterministic source-sink pair. We keep the source-sink pairs of G_1 and edges of G_2.

For simplicity, assume that each edge in Figure 1 has capacity 1. The effective capacity at each edge seen by G_1 under network coding is more than that under flows. Intuitively, replacing each edge with a source-sink pair of G_2 should give network coding a “capacity advantage” of $(1 + \epsilon_2)$ over multicommodity flow. Since the information transferred grows linearly with the capacity, the new information exchanged between source-sink pairs in the gadget under network coding should be $(1 + \epsilon_1)(1 + \epsilon_2)$ times the information exchanged under flows.

We need to be careful because G_2 exhibits a gap only when we need to send information from all sources simultaneously. We address this by adding more copies of G_1 to the graph tensor and replacing its edges with other source-sink pairs of the copies of G_2 as in Figure 3. In each copy of G_1, we replace all edges with the same source-sink pair of G_2. At the same time, each copy of G_2 serves to replace the same edge in all copies of G_1. This is done to facilitate the proof of the upper bound on the MCF rate in the resulting graph.

Our work is not done here. It is easy to get a lower bound on the network coding rate on the final graph G by just showing a network coding solution. For this, informally, we just compose the network coding solutions for G_1 and G_2. The hard part is to get an upper bound on the multicommodity flow rate. Since MCF is a linear program, we can upper bound the value of multicommodity flows by looking at the dual solution of its relaxed linear program. This dual, described in Section 2, involves computing shortest distances between source-sink pairs under some metric. This metric readily tensorizes: we can take the length of an edge in a copy of G_2 to be the product of the length of that edge in G_2 times the length of the edge(s) in G_1 this particular copy of G_2 is replacing. The problem is to get the lengths of *whole paths* to tensorize.

What could go wrong? Consider Figure 3. We would ideally want the dotted paths as in Figure 3b to be the shortest path between $S1$ and $T1$ in G', since its length is the length of the shortest $s_1 - t_1$ path in G_1 times the length of the shortest $s_{1X} - t_{1X}$ path in G_2. Unfortunately, during the tensoring operation we inadvertently introduce additional $s_1 - t_1$ paths that do not correspond to “products” of paths from G_1 and G_2. For example, the dashed path in Figure 3a is a “cheating” path which can make the distance between s_1 and t_1 shorter than expected. We deter the use of “cheating” paths by increasing the number of hops between different copies of G_1 that a path has to take before it reaches the same copy again. The technical ingredient which prevents such cheating is in the design of the bipartite graph which tells which copy of G_1 should use which copies of G_2 (and how to connect them). To prevent cheating, the bipartite graph will need to be of high girth. The crucial part of the construction is thus constructing high girth bipartite graphs while still keeping check on the size so as to get a $O(\log(\text{size})^c)$ $(c < 1)$ gap when the tensor is applied repeatedly.

Discussion

A natural question arises: Can we have a tensor construction that starts with a graph G having some gap between the multicommodity flow rate and the network coding rate and outputs a graph G' with gap $\omega(\log(|G'|))$, thus contradicting (3) and proving the Li and Li conjecture? We address this question with respect to our construction in Section 4. We show that the MCF vs. Sparsest Cut gap tensorizes for our construction, and thus the tensorization process on its own cannot cause the gap to exceed $O(\log|G'|)$.

At the same time, if one's goal is to prove the conjecture, it might be easier to reach a contradiction to the gap being $(\log|G|)^c$ than to a constant gap.

2 Preliminaries

In this section we introduce the problems that we will be interested in studying and any relevant notation. Where appropriate, we use the same notation and definitions as [1, 7].

When $G = (V, E)$ is a graph, we specify vertex set of G with $V(G)$ when the underlying graph G is not clear from the context. Similarly $E(G)$ represents the edge set for graph G. The set $\{1, 2, ..., n\}$ is represented by $[n]$. $I(G)$ denotes the set of k source-sink pairs $(s_i, t_i), i \in [k], s_i, t_i \in V$. Given a bipartite graph $B = (V_1, V_2, E)$, we denote the left side of the graph by $V_1(B)$ and the right by $V_2(B)$. A bipartite graph is (r, s) bi-regular when each vertex on the left side has degree r whereas each vertex on right side has degree s.

2.1 Network coding

▶ **Definition 3.** An instance of the k-pairs communication problem consists of

- a graph $G = (V, E)$,
- a capacity function $c : E \to \mathbb{R}^+$,
- a set I of commodities of size k, each of which can be described by a triplet of values (s_i, t_i, d_i) corresponding to the source node, the sink node and the demand of commodity i.

In line with [1], for undirected graphs we consider each edge e as two directed edges $\vec{e}, \overleftarrow{e}$, whose capacities will be defined later. It will also be convenient to think of source and sink nodes as edges. Therefore, for every source and sink pair (s_i, t_i), we create new nodes S_i, T_i and connect them via single edges to s_i, t_i respectively. These edges are of unbounded capacity and we will refer to these as the source and sink edges respectively. Every source S_i wants to communicate a message to its sink.

We give the formal definition of a network coding solution in Appendix A. Let M_i be the set of messages the i-th source-sink pair wants to communicate, and $M = \prod_i M_i$. Let $\Delta(e)$ be the alphabet of characters available at edge e. Informally, the solution to a network coding problem must specify for each edge e a function $f_e : M \to \Delta(e)$, which dictates the character transmitted on that edge. The function f_e must be computable from the characters on the incoming edges at the sender end point. The message at the source and sink edges of any commodity must agree.

The network coding rate (henceforth known as coding rate) is the largest value r such that for each source-sink pair at least $r \cdot d_i$ information is transmitted while preserving the capacity constraint on all edges.

2.2 Multicommodity flow problems and sparsity cuts

A flow problem consists of a graph $G = (V, E)$ with k commodities together with k pairs of nodes (s_i, t_i) and quantities d_i. The goal is to transmit d_i units of commodity i from s_i to t_i while keeping the total sum of commodities that go through a given edge e below its capacity $c(e)$. There are many optimization problems surrounding this problem. We will focus on the following one: what is the largest λ such that at least λ fraction of each commodity's demand is routed? This is justified by assuming that no commodity is prioritized over another and that all resources are shared. We refer to this quantity as the flow rate of the graph. There are well-known linear programming formulations for these problems (see LP 5 in Section B in the appendix). Since we will be interested in providing provable upper bounds to the flow rate, it will suffice to look at the dual of this problem. In particular, we use the variables on the following dual LP to provide upper bounds on the flow rate of the sequence of graphs we create. We will refer to the $w_{(u,v)}$ as the weight of edge (u, v) in the dual solution.

$$\begin{aligned}
&\text{minimize} \sum_{u,v} w_{(u,v)} c(u,v) && \\
&\text{subject to} \sum_{(s_i,t_i)} l(s_i,t_i) d_i \geq 1 && \text{(Distance Constraint)} \\
&\qquad \sum_{(u,v)\in p} w_{(u,v)} \geq l(s_i,t_i) && \forall i \in [k], p \in P_i \\
&\qquad w_{(u,v)} \geq 0 && \forall (u,v) \in E \wedge \forall (u,v) = (s_i,t_i)
\end{aligned} \tag{4}$$

This LP introduces a semi-metric on the graph which assigns weights to the edges. $l(s_i, t_i)$ is the shortest distance between $i-$th source-sink pair w.r.t. this metric. The goal is to minimize the weighted length of the edges of the graph while maintaining a certain separation between the source-sink pairs. Zero weight edges can be problematic for our graph tensor since they may reduce the weighted girth of the graph in ways we cannot account for. Our tensor, however, does not produce new zero weight edges. Therefore it suffices for our purposes to show that we can get rid of them at the beginning of the construction.

▶ **Lemma 4.** *If G is a graph such that the gap between the flow rate and the coding rate is $(1 + \epsilon)$, a new graph G' can be constructed such that the gap does not decrease and all the edge weights in LP (4) are non-zero.*

Proof. We defer the proof of this lemma to Section B of the appendix. ◀

Interestingly, this lemma is not true for directed graphs.

3 Construction

In this section we present the construction of our graph tensor and prove our main results, Theorems 1 and 2. The construction takes two graphs with small gaps and tensors them in such a way that the resulting graph has a gap equal to the product of the previous gaps. Iteratively tensoring a graph with a small gap with itself will yield our main results.

Throughout this section, when referring to a graph $G_i = (V_i, E_i), i \in [2]$, k_i is the number of source-sink pair, v_i the number of vertices and m_i the number of edges. The capacity of edge $e \in E_i$ will be denoted by c_{ie}.

3.1 Overview

As mentioned in Section 1, we need a bijection between the graph tensor on G_1 and G_2 and bipartite graphs. We represent the copies of G_1 by numbered nodes on the left side of the bipartite graph (say B) and copies of G_2 by nodes on the right side of B. We add an edge (i, j) in B when an edge in the i-th copy of G_1 got replaced by the j-th copy of G_2 aligned at a specific source-sink pair. But this definition of bipartite graph B loses information about which specific edge was replaced with which specific source-sink pair. Thus, we consider a variant of bipartite graphs: *colored bipartite graphs*, which have two colors associated with each edge. We will use the first color to represent the edge that got replaced in a copy of G_1 and the other to represent the source-sink pair of G_2 that replaced that edge. Thus, edges of B get colored from the set $[m_1] \times [k_2]$. Note that each vertex on the left side has degree m_1 and that on right hand side has degree k_2. The formal description of colored bipartite graphs and *graph tensor* based on this idea is given in Subsection 3.2.

As discussed in Section 1, we can avoid "cheating" paths by increasing the number of hops that a dashed path (Figure 3) needs to take to come back to the same copy of G_1. Our first requirement would be for the colored bipartite graph B to have high girth. Lemma 19 states the existence of nearly optimal sized high girth bipartite graphs and Subsection 3.2.2 shows how to construct specific colored bipartite graphs (as in Subsection 3.2) of high girth.

Is having a high girth B sufficient for the number of hops to be large? No. When G_2 has two sources at the same vertex, the end points (on source side) of the edges in copies of G_1 that these two source-sink pairs replaced will collapse on the same vertex implying that we can move between these copies of G_1 instantly without traveling along any edge in the tensored graph. But, we would have travelled two consecutive edges in B. To remedy this, we condition on the graph G_2 to have all sources and sinks lying on distinct vertices. Note that the length of the cheating paths is defined with respect to the weights of edges in a dual solution. Thus, we cannot just transfer the source/sinks to leaves at the corresponding vertex through infinite capacity edges as they would always get weight 0 in the dual. In Subsection 3.2.1, we present a way to modify graph G_2 to satisfy the above condition.

The multicommodity flow rate for the tensored graph is upper bounded by constructing a dual solution for it based on dual solutions for graphs G_1 and G_2. In Subsection 3.2.3, we show the dual construction and prove that the gap of the tensored graph is the product of the previous gaps given appropriate girth.

The last subsection of this section contains the details of repeated tensoring to get Theorem 1.

3.2 Graph Tensor

▶ **Definition 5. Colored Bipartite Graph:** We define $\mathcal{B}_{n_1,n_2,d_1,d_2,g,q_1,q_2}$ to be the set of bipartite graphs (V_1, V_2, E) with girth g, $|V_1| = n_1, |V_2| = n_2$, such that degree of each vertex in V_1 and V_2 is d_1 and d_2 respectively and each edge is given a color l_e in $[q_1] \times [q_2]$. Note that $n_1 d_1 = n_2 d_2$.

▶ **Definition 6.** $\boldsymbol{T(G_1, G_2, B)}$ is defined to be the graph tensor on directed graphs G_1 and G_2 based on the colored bipartite graph B.

For $T(G_1, G_2, B)$ to be defined, we need B to satisfy the following properties:

1. $B \in \mathcal{B}_{n_1,n_2,m_1,k_2,g,m_1,k_2}$ for some $n_1, g \in \{1, 2, ...\}$.
   - G_1 has m_1 edges and G_2 has k_2 source-sink pairs. Therefore the degrees of each node on the left and right sides should be m_1 and k_2, respectively.
   - As mentioned in Subsection 3.1, edges must be colored in the set $[m_1] \times [k_2]$.
2. $\forall v \in V_2$, the set $B_v = \{b_e \mid e \text{ is incident to } v \text{ and } l_e = (a_e, b_e)\}$ is the complete set $[k_2]$. We want each source-sink pair of a copy of G_2 to replace some edge in a copy of G_1.
3. $\forall u \in V_1$, the set $A_u = \{a_e \mid e \text{ is incident to } u \text{ and } l_e = (a_e, b_e)\}$ is the complete set $[m_1]$. This ensures that each edge in a copy of G_1 is replaced.
4. $\forall v \in V_2$, the set $A_v = \{a_e \mid e \text{ is incident to } v \text{ and } l_e = (a_e, b_e)\}$ has cardinality 1. To define capacities in the new tensored graph naturally, we want that each source-sink pair in a copy of G_2 replaces some unique edge in its corresponding copy of G_1.
5. $\forall u \in V_1$, the set $B_u = \{b_e \mid e \text{ is incident to } u \text{ and } l_e = (a_e, b_e)\}$ has cardinality 1. This ensures that each edges in a copy of G_1 is replaced by the same source-sink pair in different copies of G_2.

We construct the graph $T(G_1, G_2, B)$ as follows:

- Enumerate the n_1 nodes in $V_1(B)$ and n_2 nodes in $V_2(B)$: $u^{(1)}, u^{(2)}, ..., u^{(n_1)}$ and $v^{(1)}, .., v^{(n_2)}$ respectively.
- Enumerate all the edges in G_1: $e^{(1)}_{G_1}, e^{(2)}_{G_1}, ..., e^{(m_1)}_{G_1}$.
- Create n_1 copies of G_1 (vertices and source-sink pairs) and n_2 copies of G_2 (vertices and edges). Represent the x^{th} copy of graph $G_y, y \in \{1, 2\}$ by $G^{(x)}_y$. Let $u^{(i)} \in V_1(B)$ represent the i-th copy of G_1 and $v^{(j)} \in V_2(B)$ represent the j-th copy of G_2.
- For every edge $e = (u^{(i)}, v^{(j)})$ colored (p, k), merge the vertices $a_{G^{(i)}_1}$ and $s_{kG^{(j)}_2}$, and $t_{kG^{(j)}_2}$ and $b_{G^{(i)}_1}$ in $T(G_1, G_2, B)$. Here, $e^{(p)}_{G^{(i)}_1} = (a_{G^{(i)}_1}, b_{G^{(i)}_1})$ is the p^{th} edge in the i-th copy of G_1 and $(s_{kG^{(j)}_2}, t_{kG^{(j)}_2})$ is the k^{th} source-sink pair of the j^{th} copy of G_2. Informally, we are replacing each edge in a copy of G_1 by a copy of G_2 with end points aligned with the k^{th} source-sink pair. Set the capacity of every edge e' in this j^{th} copy of G_2 to be $c_{1e^{(p)}_{G_1}} c_{2e'}$. This can be done consistently due to Property (4).
- Make all the edges undirected.

We define a tensor on directed graphs to allow for composition of network coding solutions of G_1 and G_2. The direction of an edge in G_1 tells us how to align the source-sink pair of G_2 on that edge. An example of a tensor is the graph in Figure 3.

3.2.1 Standard Forms and Graph Extensions

Without loss of generality, we assume that for the graph G, all the demands $d_i, i \in [|I(G)|]$ are equal. Otherwise, we can just divide the demands into small demands of size x such that x divides all the initial rational demands. As discussed in Subsection 3.1, we want all sources and sinks to lie on distinct vertices. For all the dual solutions D that we mention, we assume that D does not contain any zero weight edges. This is justified by Lemma 4 and the fact that new duals constructed while tensoring, which will be defined later, don't create zero weight edges. We say a graph-dual pair (G, D) is in *standard form* when all the assumptions above are satisfied.

We now present a construction whose goal is to make all $s_i, t_i, i \in [k]$ lie on distinct vertices.

▶ **Definition 7.** Given a graph $G = (V, E)$ with all demands being equal to d, and a dual solution D with $\frac{NC_G}{z(D)} \geq 1 + \varepsilon$, $\forall \alpha, 0 < \alpha < \varepsilon$, construct a new graph G_α such that all

$s_i, t_i, i \in [k]$ lie on distinct vertices and G_α has a dual solution $D_\alpha(G)$ with $\frac{NC_{G_\alpha}}{z(D_\alpha(G))}$ being at least $\frac{1+\varepsilon}{1+\alpha}$. G_α is defined as the **α-Extension of G given D**. $z(D)$ is the objective value of dual solution D.

Here, we just move the sources/sinks at a vertex to the leaves of the new edges added at this vertex while keeping edge capacities and dual weights in check. The detailed description of G_α and $D_\alpha(G)$ is given in Section C of the Appendix.

3.2.2 Colored Bipartite Graph Construction

We need small, colored bipartite graphs for every degree and girth to define the graph tensor on any two graphs with gaps. We construct such graphs using biregular bipartite graphs with high girth. The following lemma states the existence of nearly-optimal sized colored bipartite graphs.

▶ **Lemma 8.** *$\forall r, s, g \geq 3$, there exists a colored bipartite graph $C_{rsg} \in \mathcal{B}_{n_1,n_2,r,s,2g,r,s}$ with $n_1, n_2 \leq (9rs)^{g+3}$.*

Proof. We defer the detailed construction and proof of the next lemma to Section D of the Appendix. ◀

3.2.3 Gap Amplification

We are given G_1 and G_2 in standard form with G_y, $y \in [2]$ having gap $(1+\varepsilon_y)$. Let N_y be the optimal network coding solution for $G_y, y \in [2]$. Construct a directed graph G_1' from G_1 by replacing each (undirected) edge $e = (u, v) \in E(G_1)$ of capacity c_{1e} with 2 directed edges (u, v) and (v, u) of capacities c_{1eu} and c_{1ev} respectively. Here, c_{1eu} and c_{1ev} are the capacities of edge e used by N_1 in the defined directions. Note that $c_{1eu} + c_{1ev} \leq c_{1e}$. Without loss of generality, assume $c_{1eu} + c_{1ev} = c_{1e}$, as we can always increase one of the capacities without changing the network coding solution to get the equality. Similarly, construct G_2' from G_2 based on N_2. G_1' and G_2' has $m_1' = 2m_1$ and $m_2' = 2m_2$ edges respectively.

▶ **Definition 9.** $\mathbf{Tensor(G_1, G_2, D_1, D_2)}$ is defined as $T(G_1', G_2', B')$, where $B' = C_{m_1' k_2 g}$, $2g = \frac{2l_1 l_2}{w_1 w_2}$. Here, l_1 and l_2 are the maximum dual distances between any source-sink pair in the dual solutions D_1 of G_1 and D_2 of G_2 respectively. $w_1 > 0$ and $w_2 > 0$ are the minimum edge weights in the dual D_1 and D_2 respectively.

Define $\mathbf{Dual(G_1, G_2, D_1, D_2)}$ to be the specific dual solution for Tensor(G_1, G_2, D_1, D_2) that would be constructed in proof of Lemma 12.

All the demands in the graph Tensor(G_1, G_2, D_1, D_2) are equal to $\frac{d_1 d_2}{q}$ where $q = \frac{V_1(B')}{k_2} = \frac{V_2(B')}{m_1}$. Here, d_y is the demand of each commodity in graph $G_y, y \in [2]$. We use such a scaling to have a simple description of Dual(G_1, G_2, D_1, D_2) in terms of D_1 and D_2.

We prove the gap amplification part of Theorem 2 next. The details of how size grows are in the next subsection.

▶ **Theorem 10.** *Given graphs G_1 and G_2 in standard form and dual solutions D_1 and D_2 respectively, such that $\frac{NC_{G_y}}{z(D_y)} \geq 1 + \varepsilon_y, y \in [2]$, $G =$ Tensor(G_1, G_2, D_1, D_2) has a dual solution $D =$ Dual(G_1, G_2, D_1, D_2) such that $\frac{NC_G}{z(D)} \geq (1+\varepsilon_1)(1+\varepsilon_2)$.*

In the next two lemmas, we lower bound the network coding rate and upper bound the multicommodity flow rate of G.

▶ **Lemma 11.** *The coding rate for G is at least $r_1 r_2(1+\varepsilon_1)(1+\varepsilon_2)q$ where r_1 and r_2 are objective values of dual solutions D_1 and D_2 respectively.*

Proof Sketch. The proof follows from composing the optimal network coding solutions of G_1 and G_2. The details are given in Section E of the Appendix. ◀

▶ **Lemma 12.** *D has objective value at most $r_1 r_2 q$ where r_1 and r_2 are the objective values of dual solutions D_1 and D_2 respectively.*

Proof Sketch. $G = T(G'_1, G'_2, B')$ where variables are as defined in Definition 9. For every edge $e \in E(G)$, e is the undirected version of an edge in a copy of G'_2 (of say e_2 in G'_2) and this copy of G'_2 must have replaced a unique edge (say $e_1 \in E(G'_1)$) in different copies of G'_1. Edges e_1 and e_2 are directed edges but have undirected counterparts in G_1 and G_2. Let w_{1e_1} and w_{2e_2} be the weights given to the counterpart edges of e_1 and e_2 in dual solutions D_1 and D_2 respectively. Give weight $w_e = w_{1e_1} w_{2e_2}$ to edge e in D. Note that $\forall e, w_e > 0$ if $w_{1e_1}, w_{2e_2} > 0 \forall e_1, e_2$. Thus, non-zero dual solutions D_1 and D_2 give a non-zero dual solution D to graph G. We still need to show that D is a valid dual solution for G. Since B' has girth at least $\frac{2l_2 l_1}{w_1 w_2}$ and G_2 is in standard form, the dotted paths (as in Figure 3) are the shortest paths with respect to dual D. We can then write the distances between source-sink pairs in G in terms of the distance of this source-sink pair in G_1 w.r.t. D_1 and the distance of the source-sink pair in G_2 that replaced edges in this copy of G_1 w.r.t. D_2. This allows us to easily show the satisfiability of the distance constraint for D when demands are as specified in Definition 9.

The detailed proof is continued in Section E of the Appendix. There we also show $z(D) = \frac{n_1}{k_2} z(D_1) z(D_2) = q r_1 r_2$. ◀

Proof of Theorem 10. It follows from just dividing the lower bound on the network coding rate of G and the upper bound on the objective value of D obtained in Lemma 11 and Lemma 12. ◀

In the next subsection, we show how to repeatedly apply this construction. Note that, we can only apply the tensor construction on graphs in *standard form.* The following lemma allows us to tensor the new graph obtained with itself.

▶ **Lemma 13.** *Given G_1 and G_2 in standard form, Tensor(G_1, G_2, D_1, D_2) is also in standard form.*

Proof. We defer the proof of this lemma to Section E of the Appendix. ◀

3.2.4 Iterative Tensoring

In the next two statements size refers to the number of vertices in the graph A_i. The calculation of the size involves calculating the required girth at each iteration and the size of the colored bipartite graph used to tensor at each iteration.

▶ **Theorem 14.** *Given a graph $A = (V, E)$ with gap $(1+\varepsilon)$, we can construct a sequence of graphs $A_i = (V_i, E_i)$ with gap at least $(1+\frac{\varepsilon}{2})^{2^i}$, size at most $(3c_m)^{(4c_1)^{2^{i+1}}}$ where c_m and c_1 are absolute constants.*

Proof. We defer the proof to Section F of the Appendix. Let $\alpha = \frac{1+\varepsilon}{1+\varepsilon/2} - 1$. The proof first considers the α-Extension of A to start the recursion with a graph in standard form, then recursively defines pairs of tensored graphs and duals (A_i, D_i) such that the gap increases geometrically. ◀

Proof of Theorem 1. Now, we calculate an expression for the gap in terms of size. $\frac{\log(\text{gap})}{\log(1+\varepsilon/2)} \geq 2^i \geq \frac{\log\log(\text{size})-\log\log 3c_m}{\log(16c_1^2)}$. Thus, we get a sequence of graphs with gap at least $\Omega((\log(\text{size}))^{c_2})$ where c_2 is an absolute positive constant less than 1 equal to $\frac{\log(1+\varepsilon/2)}{\log(16c_1^2)}$. ◀

4 Limits of the Construction

In this section, we show that the construction we present can not be used "as is" to prove the Li and Li conjecture. The requirement for the underlying bipartite graph to have a high girth seems to heavily contribute to the size of the graph in the next iteration. Can we do better in terms of size to yield a gap of $\omega(\log|G|)$ by choosing a smaller bipartite graph at every iteration while still having a clever upper bound on the multicommodity flow in the new graph? The answer is no. Theorem 15 states that for every colored bipartite graph B, the tensor of G_1 and G_2 with B as basis has sparsity of at least the product of the sparsities of G_1 and G_2 when the demands are all 1 in all the graphs. With the appropriate demands, this means that the sparsity grows exactly like the coding rate as in Lemma 11. Thus, for any iterative tensoring procedure that starts with a graph G with NC/MCF gap and repeatedly tensors the graph at the ith iteration (G_i) with itself or with G based on a colored bipartite graph B_i will end up with a graph G' with $\omega(\log|G'|)$ gap. Hence we can start with a graph H with a gap between the flow rate and the sparsity and apply this procedure to get a graph H' with $\omega(\log|H'|)$ gap between the flow rate and the sparsity, contradicting the bounds from [10]. This means that through iterative tensoring, if we were able to prove the conjecture, we would also prove the statement that there exists no graphs with sparsity-multicommodity flow rate gap which is clearly false.

▶ **Theorem 15.** *For any G_1, G_2, B for which $G = T(G_1', G_2', B)$ is defined (G_1' and G_2' are directed graphs obtained from G_1 and G_2 by directing each edge arbitrary in two directions such that new capacities add up to the previous),*

$$\textit{Sparsity}(G) \geq \textit{Sparsity}(G_1) \cdot \textit{Sparsity}(G_2).$$

when the demands of G_1, G_2 and G are all scaled to 1.

Proof. We defer the proof to Section G in the Appendix. ◀

Note that this gives an identical theorem as Theorem 10 for sparsity vs multicommodity flow rate for the corresponding demands.

References

1. M. Adler, N. J. A. Harvey, K. Jain, R. Kleinberg, and A. Rasala Lehman. On the capacity of information networks. In *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithm*, SODA '06, pages 241–250, Philadelphia, PA, USA, 2006. Society for Industrial and Applied Mathematics.
2. Rudolf Ahlswede, Ning Cai, S-YR Li, and Raymond W Yeung. Network information flow. *IEEE Transactions on information theory*, 46(4):1204–1216, 2000.
3. R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows: Theory, Algorithms, and Applications.* Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1993.
4. Anna Blasiak, Robert Kleinberg, and Eyal Lubetzky. Lexicographic products and the power of non-linear network coding. In *Foundations of Computer Science (FOCS), 2011 IEEE 52nd Annual Symposium on*, pages 609–618. IEEE, 2011.

5 C. Fragouli and E. Soljanin. Network coding applications. *Found. Trends Netw.*, 2(2):135–269, January 2007.

6 Z. Furedi, F. Lazebnik, A. Seress, V. A. Ustimenko, and A. J Woldar. Graphs of prescribed girth and bi-degree. *Journal of Combinatorial Theory, Series B*, 64(2):228–239, 1995.

7 N. J. A. Harvey, R. Kleinberg, and A. Rasala Lehman. Comparing network coding with multicommodity flow for the k-pairs communication problem, 2004.

8 K. Jain, V. V. Vazirani, and G. Yuval. On the capacity of multiple unicast sessions in undirected graphs. *IEEE/ACM Trans. Netw.*, 14(SI):2805–2809, June 2006.

9 G. Kramer and S. A. Savari. Edge-cut bounds on network coding rates. *J. Netw. Syst. Manage.*, 14(1):49–67, March 2006.

10 T. Leighton and S. Rao. Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *J. ACM*, 46(6):787–832, November 1999.

11 Z. Li and B. Li. Network coding in undirected networks, 2004.

12 Z. Li, B. Li, D. Jiang, and L. C. Lau. On achieving optimal throughput with network coding. In *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies, 13-17 March 2005, Miami, FL, USA*, pages 2184–2194, 2005.

13 M. Medard and A. Sprintson. *Network Coding: Fundamentals and Applications*. Academic Press. Elsevier, 2012.

14 R. W. Yeung. *Information theory and network coding*. Springer Science & Business Media, 2008.

15 R. W. Yeung, S.-Y. R. Li, N. Cai, and Z. Zhang. Network coding theory: Single sources. *Commun. Inf. Theory*, 2(4):241–329, September 2005.

A Definition of Network Coding

Let $M(i)$ be the set of all messages s_i wants to send, and let $M = \Pi_i M(i)$. For every $v \in V$, let $\text{In}(v) \subseteq E$ denote the set of edges incident to e.

▶ **Definition 16.** A *network coding solution* for a graph G specifies for each directed edge $e \in E$ an alphabet $\Gamma(e)$ and a function $f_e : M \to \Gamma(e)$ mapping the symbol transmitted on edge e. This must satisfy the following two conditions:

- Correctness: each sink node receives the message from its corresponding source, i.e. $f_{T(i)} = f_{S(i)}$.
- Causality: every message transmitted on edge e is computable from information received at its tail vertex at a time prior to the message's transmission.

▶ **Definition 17.** A *causal computation* of a network consists of

- A sequence of edges $e_1, ..., e_T$ where each edge can appear multiple times,
- A sequence of alphabets $\Lambda_1, ..., \Lambda_T$, and
- A sequence of coding functions $\rho_1, ..., \rho_T$, which in turn satisfy:
  1. For each function ρ_t such that $e_t = (u, v)$ is not a source edge, the value of ρ_t is uniquely determined by the values of the functions in the set $\{\rho_x : x < t, e_x \in \text{In}(u)\}$.
  2. For each edge e, the Cartesian product of the alphabets in the set $\{\Lambda_i : e_i = e\}$ is equal to $\Gamma(e)$.
  3. For each edge e, the set of coding functions $\{\rho_i : e_i = e\}$ together define the coding function f_e specified by the network coding solution.

At this point we are equipped with the tools needed to define the network coding rate, the information-theoretic equivalent of the flow rate.

▶ **Definition 18.** A *network coding solution* for a graph G achieves a rate r if there exists a constant $b \geq 0$ such that

- $H(S(i)) \geq r \cdot d_i \cdot b$ for each commodity i,
- for each edge $e \in E$, $H(\vec{e}) + H(\overleftarrow{e}) \leq c(e) \cdot b$,

where by $H(\vec{e})$ we denote the entropy of edge $\vec{e}$. The coding rate is defined to be the supremum of the rates of all network coding solutions.

B Multicommodity Flows

The standard LP formulation for concurrent multicommodity flow problems is written below. It has a variable for every path $p \in P_i$, where P_i is the set of all paths between s_i and t_i. We want to find the largest rate λ that can be concurrently sent between all source-sink pairs subject to the path variables being non-negative and not exceeding the capacity of any edge over all commodities.

$$\begin{aligned} \text{maximize} \quad & \lambda \\ \text{subject to} \quad & \sum_{p \in P_i} f(p) \geq \lambda d_i \quad && \forall i \in [k] \\ & \sum_{p : e \in p} f(p) \leq c(e) && \forall e \in E \\ & f(p) \geq 0 && \forall p \\ & \lambda \geq 0 \end{aligned} \tag{5}$$

Proof of Lemma 4. We contract all the edges with zero weight in the dual. We need to show that the gap does not decrease. Removing a zero dual variable from a multicommodity solution cannot improve the flow rate, since the distances and the dual objective remains the same. We can use the same coding solution for the new graph with the exception that we now compose the encoding on the edges that were contracted. This shows that the flow rate does not increase and the coding rate does not decrease, proving that their ratio does not decrease. ◀

C Standard Form

This section gives the detailed description of G_α and $D_\alpha(G)$. Let k_v be the number of sources and sinks at vertex $v \in V(G)$. In the graph G_α, add k_v edges (leaves) at v with capacity $z(D)d(1+\varepsilon)$ and shift all the sources or sinks at v to the unique endpoints of these leaves. As each source sends $\geq z(D)d(1+\varepsilon)$ amount of information in an optimal network coding solution and can still send $z(D)d(1+\varepsilon)$, the network coding rate doesn't decrease below $z(D)(1+\varepsilon)$. We construct $D_\alpha(G)$ as follows:

1. For each edge originally in G, assign the same weights as in D.
2. Give weight $\frac{\alpha}{kd(1+\varepsilon)}$ to the new edges.

Distances in the dual don't decrease, so $D_\alpha(G)$ is a valid solution. Since we added k new edges, $z(D_\alpha(G)) = kz(D)d(1+\varepsilon)\frac{\alpha}{kd(1+\varepsilon)} + z(D) = z(D)(1+\alpha)$. Thus, $\frac{NC_G}{z(D\alpha(G))} \geq \frac{1+\varepsilon}{1+\alpha}$.

D Colored Bipartite Graph Construction

In this section, we give a construction for a colored bipartite graph C_B in $\mathcal{B}_{n_1,n_2,r,s,2g,r,s}$ $\forall r, s, g \geq 3$ with $n_1, n_2 \leq rs(9rs)^{g+2}$. We start with a (r,s)-biregular bipartite graph with girth at least $2g$.

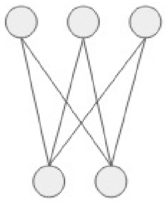


Figure 4 (2,3) biregular bipartite graph with girth 4.

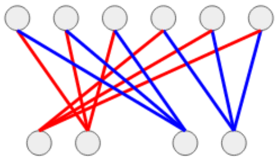


Figure 5 Intermediate graph when one set of colors hasf been assigned.

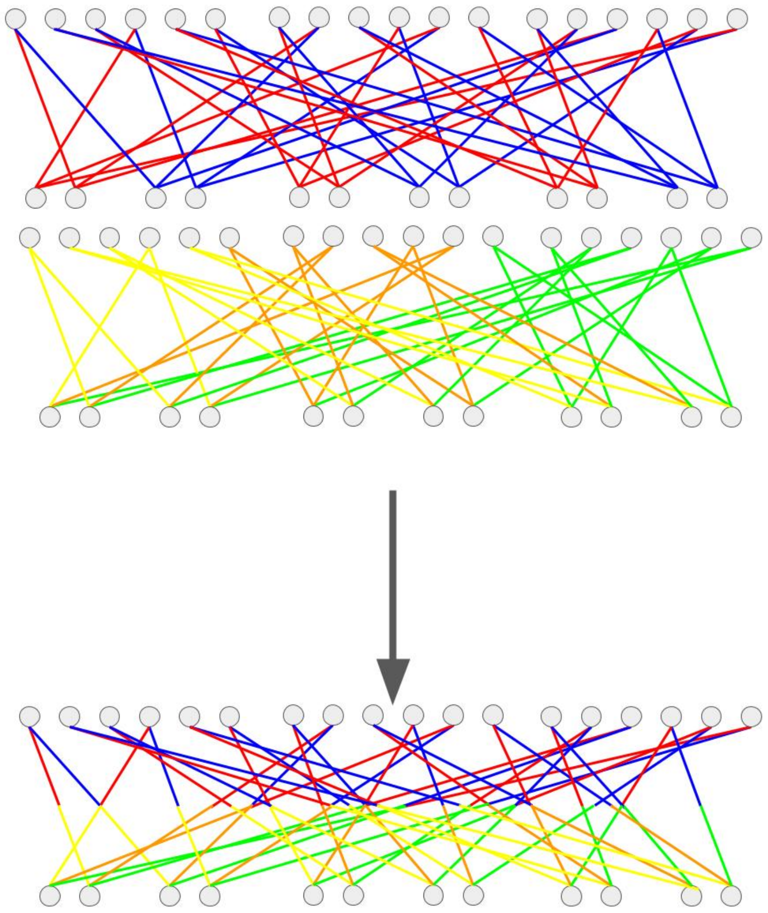


Figure 6 Final colored bipartite graph with girth 4.

▶ **Lemma 19.** *[6] For all $r, s, g \geq 3$, there exists a (r, s)-biregular bipartite graph with girth at least $2g$ and having at most $n = (9rs)^{g+2}$ vertices.*

This lemma follows from Theorem E in [6].

Proof of Lemma 8. Let $B(r, s, g)$ be a graph satisfying the above property. For simplicity, denote $B(r, s, g)$ by just $B = (V_1, V_2, E)$. Denote the coloring for every edge e by (a_e, b_e). First we construct an intermediate graph H in $\mathcal{B}_{n'_1,n'_2,r,s,2g,1,s}$ $(n'_1 = s|V_1|, n'_2 = s|V_2|)$ as follows:

1. Enumerate all the edges incident to a vertex $v \in V_2$ as $e_v^{(1)}, ..., e_v^{(s)}$.
2. Add s copies of V to graph H. Enumerate these copies as $(V_1^{(1)}, V_2^{(1)}), ..., (V_1^{(s)}, V_2^{(s)})$.
3. $\forall v^j \in V_2^{(j)}, j \in [s]$, $\forall i \in [s]$, corresponding to edge $e_v^{(i)} = (u, v) \in E(B)$, add an edge e from v^j to $u^{(((j+i-2) \bmod s)+1)} \in V_1^{(((j+i-2) \bmod s)+1)}$ (copy of u in $(((j+i-2) \bmod s)+1)$-th copy of V_1). Set $b_e = ((j+i-2) \bmod s) + 1$. Therefore, $\forall u^j \in V_1^{(j)}$, edges e' incident at u^j have $b_{e'} = j$ (same color).

For a vertex $v \in V_2^{(j)}$, the edge corresponding to $e_v^{(i)}$ comes from a vertex in $V_1^{(((j+i-2) \bmod s)+1)}$. Thus, all edges incident to v have distinct colors.

We still need to show that the girth of H is at least $2g$. For this, we show that a cycle C of length c in H implies a cycle of length $\leq c$ in B. As all the edges incident to a vertex in H correspond to different edges in B, when we project back C to a cycle C' in B, no two consecutive edges in C' are the same implying C' has no cycle of length 2. Thus, C' must have a cycle of length $3 \leq c' \leq c$. B has girth at least $2g$, so the girth of H cannot be smaller.

Now, we repeat the process for $H = (H_1, H_2)$ to get graph C_B with H_1 playing the role of V_2 and H_2 playing the role of V_1 in the above algorithm. This time we assign $a_e \in [r]$ and make r copies of H. We can see that as was the case for b_e, each vertex in a copy of H_1 gets r distinct a_e values and each vertex in a copy of H_2 gets the same a_e depending on which copy it belongs to. The girth doesn't decrease on going from H to C_B giving us the result we claim. ◀

An example of a colored bipartite graph in $\mathcal{B}_{12,18,3,2,4,3,2}$ is given in Figure 6. We start with $K_{2,3}$ as in Figure 4 with girth 4. Then, we construct the intermediate graph in $\mathcal{B}_{4,6,3,2,4,1,2}$ as shown in Figure 5. The color of the edge depends on the copy it is incident to on the lower side. For a vertex on the upper side, we send edges to correct vertices in distinct copies cyclically.

E Gap Amplification Proofs

Proof of Lemma 11. Graph G_1 has a network coding rate of at least $r_1(1+\varepsilon_1)$ and hence each source sends $r_1(1+\varepsilon_1)d_1$ amount of information to its corresponding sink, and similarly for G_2. This is true even for the directed graphs G'_1 and G'_2 by definition. While constructing $T(G'_1, G'_2, B')$, we aligned the source-sink pair in the same direction as the directed edge. This allows us to compose the network coding solutions (N_2 over N_1) to get the information sent from each source in G to be at least $r_1r_2(1+\varepsilon_1)(1+\varepsilon_2)d_1d_2$. This is due to the fact that as we are replacing edges in G'_1 by a source-sink pair of a copy of G'_2, the effective capacity seen by the replaced edge (e) with capacity c_{1e} is now $c_{1e}r_2(1+\varepsilon_2)d_2$. Thus, the coding rate for graph G is at least $\frac{r_1r_2(1+\varepsilon_1)(1+\varepsilon_2)d_1d_2}{(\text{demand in graph } G)} = \frac{r_1r_2(1+\varepsilon_1)(1+\varepsilon_2)d_1d_2q}{d_1d_2} = r_1r_2(1+\varepsilon_1)(1+\varepsilon_2)q$. ◀

Proof of Lemma 12. Here, we prove that D is indeed a valid dual solution. B' has $n_1 = k_2q$ nodes on the left side. Let $l_1(s_i, t_i)$ denote the shortest distance between i-th source-sink pair with respect to dual D_1. Let $l_2(s_i,t_i)$ denote the shortest distance between i-th source-sink pair with respect to dual D_2.

Let $G_1'^u$ and $G_2'^u$ be the undirected version of the graphs G_1' and G_2' respectively. $G_1'^u$ and $G_2'^u$ are graphs G_1 and G_2 where each edge is divided into 2 edges with capacities adding up to the previous one. Construct dual solutions D_1' and D_2' for $G_1'^u$ and $G_2'^u$ such that each divided edge still gets the same weight as in dual solutions D_1 and D_2. The distances between source-sink pairs remain the same. In G, calculate the shortest distance, i.e. $l(s_i^{(y)},t_i^{(y)})$ between source-sink pair $(s_i^{(y)},t_i^{(y)})$ which corresponds to the i-th source-sink pair (s_i,t_i) in the y-th copy of $G_1'^u$ (finally, we make the graph undirected). In this copy of $G_1'^u$, assume that we replaced each edge with the j_y-th source-sink pair of $G_2'^u$ (this is unique due to Property (2) in Definition 6). Therefore, according to D, $l(s_i^{(y)},t_i^{(y)}) \le l_1(s_i,t_i)l_2(s_{j_y},t_{j_y})$ (these correspond to dotted paths). Any other path from $s_i^{(y)}$ to $t_i^{(y)}$ involves traversing to another copy of $G_1'^u$ through a copy of $G_2'^u$ that replaced edges in this copy of $G_1'^u$. This transition from a copy of $G_1'^u$ to another copy of $G_1'^u$ in G corresponds to two consecutive edges in the bipartite graph B'. Any such path in G having no loops would thus have to make at least g of these transitions to revert back to the original copy of $G_1'^u$ containing the source. Here, the girth of graph B' is at least $2g$. Each transition involves crossing at least one edge (in a copy of G_2') with weight at least w_1w_2 in D because G_2, being in standard form, has all sources and sinks lying on distinct vertices and vertices of a copy of G_1' connect only to the vertices of a copy of G_2' carrying a unique source or sink. Thus, such a path would have distance at least $gw_1w_2 = \frac{l_2l_1}{w_1w_2}w_1w_2 \ge l_1l_2$ using l_1,l_2,w_1,w_2 from Definition 9. The cheating paths have distance at least l_1l_2 implying $l(s_i^{(y)},t_i^{(y)}) = l_1(s_i,t_i)l_2(s_{j_y},t_{j_y})$. The left hand side of the distance constraint in LP 4 becomes $\sum_{i=1,y=1}^{k_1,n_1}\frac{d_1d_2}{q}l(s_i^{(y)},t_i^{(y)}) \ge 1$, where the first expression in the summand is the demand of source-sink pairs in G. Now, we are going to show that the constraint is true:

$$\sum_{i=1,y=1}^{k_1,n_1}\frac{d_1d_2}{q}l(s_i^{(y)},t_i^{(y)}) = \sum_{i=1,y=1}^{k_1,n_1}\frac{d_1d_2}{q}l_1(s_i,t_i)l_2(s_{j_y},t_{j_y}) = \frac{d_1d_2}{q}\sum_{y=1}^{n_1}l_2(s_{j_y},t_{j_y})\sum_{i=1}^{k_1}l_1(s_i,t_i)$$

$$= \frac{1}{q}\cdot\frac{n_1}{k_2}\left(\sum_{j=1}^{k_2}d_2l_2(s_j,t_j)\right)\left(\sum_{i=1}^{k_1}d_1l_1(s_i,t_i)\right) \ge \frac{n_1}{qk_2} = 1$$

The second to last equality follows from the fact that there are total n_1 copies of G_1', j_y is fixed for fixed y-th copy of G_1' and each $l_2(s_j,t_j)$ ($j \in [k_2]$) is thus counted $\frac{n_1}{k_2}$ time. The last inequality follows from D_1' and D_2' being valid dual solutions of $G_1'^u$ and $G_2'^u$ respectively (distance constraints). This proves that D is a valid dual solution.

The value of $z(D_1')$ for graph $G_1'^u$ is r_1. D_1' assigns the same dual weights as that of D_1 for the divided edges and is a valid dual solution for G_1', and similarly for D_2'. We can see from the construction of D and the edge capacities that $z(D) = \frac{n_1}{k_2}z(D_1')z(D_2') = qr_1r_2$. D is a function of G_1, G_2, D_1, D_2. ◀

Proof of Lemma 13. Demands are equal for all source-sink pairs in $G = \text{Tensor}(G_1,G_2,D_1,D_2)$ by definition. We need to prove that all sources and sinks in G still lie on distinct vertices. We don't add any new source-sink pairs and thus, each source-sink pair lies on distinct vertices on a copy of G_1'. While constructing $T(G_1',G_2',B')$, we merge a vertex v in a copy of G_1' with a source or a sink vertex of a copy of G_2' and since each vertex contains a unique source or sink of G_2', no two vertices from different copies of G_1' are merged together. This implies that all sources and sinks still lie on distinct vertices of G. ◀

F Proof of Theorem 14

Proof. Using Lemma 4, we can assume that graph A has an optimal dual solution D with all dual variables being non-zero. It is without loss of generality that A has equal demands for all source-sink pairs. Define A^* to be the α-Extension of A given D and $D^* = D_\alpha(A)$ $(1+\alpha = \frac{1+\varepsilon}{1+\varepsilon/2})$. Let A^* have c_n vertices, c_m edges and c_k source-sink pairs having c_d demand each. Without loss of generality we can assume that $c_m \geq c_k, c_n$ as otherwise we can just divide some edges into multiple edges with reduced capacities. Let l be the largest distance between any source-sink pair in the dual D^* and $w > 0$ be the minimum weight of an edge in dual D^*. We also know that $\frac{NC_{A^*}}{z(D^*)} \geq \frac{1+\varepsilon}{1+\alpha} = 1 + \frac{\varepsilon}{2}$. As the objective value of any dual solution is at least the flow rate, we get that A^* has a gap of at least $(1+\frac{\varepsilon}{2})$. A^* is in standard form. A_i is defined iteratively as follows:

$A_0 = A^*, D_0 = D^*, \varepsilon_0 = \frac{\varepsilon}{2}$.
For $i \geq 1$:
ε_i is such that $(1+\varepsilon_i) = (1+\varepsilon_{i-1})^2$.
$A_i =$Tensor$(A_{i-1}, A_{i-1}, D_{i-1}, D_{i-1})$.
$D_i =$Dual$(A_{i-1}, A_{i-1}, D_{i-1}, D_{i-1})$.

Note that $\forall i, A_i$ is in standard form using Lemma 13 and thus iterative tensoring is valid. Through Theorem 10, we know that if $\frac{NC_{A_{i-1}}}{z(D_{i-1})} \geq (1+\varepsilon_{i-1})$, then $\frac{NC_{A_i}}{z(D_i)} \geq (1+\varepsilon_{i-1})^2 = 1+\varepsilon_i$. As $\frac{NC_{A^*}}{z(D^*)} = 1 + \frac{\varepsilon}{2}$, we get $\frac{NC_{A_i}}{z(D_i)} \geq 1 + \varepsilon_i = (1+\varepsilon/2)^{2^i} \forall i$ by induction. The objective value of any dual solution is at least the flow rate implying that the gap between coding and flow rate for A_i is at least $(1+\frac{\varepsilon}{2})^{2^i}$.

To see how the size of A_i grows, we first calculate the required girth $(2g_i)$ at each iteration. From the construction of $D_i = \text{Dual}(A_{i-1}, A_{i-1}, D_{i-1}, D_{i-1})$ in the proof of Lemma 12 we see that $w_i = w_{i-1}^2, l_i \leq l_{i-1}^2$. By induction, we have that for all i, $w_i = w^{2^i}$ and $l_i \leq l^{2^i}$. From Definition 9, we have that $g_i = \frac{l_{i-1}^2}{w_{i-1}^2} \leq \frac{(l^{2^{i-1}})^2}{(w^{2^{i-1}})^2} = (\frac{l}{w})^{2^i}$. Therefore, $g_i \leq (\frac{l}{w})^{2^i} \forall i \geq 1$. Let $c = \frac{l}{w} \geq 1$.

Now, we establish an upper bound on the size of the graph. Recall A_i is the $T(A'_{i-1}, A'_{i-1}, B_i)$ where $B_i = C_{m'_{i-1}k_{i-1}g_i}$ and $m'_{i-1} = 2m_{i-1}$. A'_{i-1} is the directed graph constructed according to the optimal network coding solution of A_{i-1}. Let $n_{1i} = |V_i(B_i)|, n_{2i} = |V_2(B_i)|$. From Lemma 8, $n_{1i} \leq (9m_{i-1}k_{i-1})^{g_i+3} \leq (9m_{i-1}k_{i-1})^{c^{2^i}+3}$.

Note that $m_i = \frac{n_{1i}}{k_{i-1}} m_{A'_{i-1}} m_{A'_{i-1}} = \frac{n_{1i}}{k_{i-1}}(4m_{i-1}^2)$ and $k_i = n_{1i}k_{i-1}$. Each edge in A'_{i-1} is replaced by a copy of A'_{i-1} and each copy is counted k_{i-1} times implying $v_i \leq 2m_{i-1}v_{i-1}\frac{n_{1i}}{k_{i-1}}$.

Moreover, $\frac{m_i}{k_i} = 4(\frac{m_{i-1}}{k_{i-1}})^2$. By induction, $k_i \leq m_i$ as $c_k \leq c_m$. Likewise, we get that $\frac{m_i}{v_i} = 2\frac{m_{i-1}}{v_{i-1}} \geq 1 \forall i$. The upper bound on m_i (the number of edges in A_i) is as follows: $m_i \leq 4n_{1i}(m_{i-1}^2) \leq 4m_{i-1}^2(9m_{i-1}k_{i-1})^{g_i+3} \leq (9m_{i-1}^2)^{g_i+4} = (3m_{i-1})^{2c^{2^i}+8} \leq (3m_{i-1})^{2(c+1)^{2^i}+8} \leq (3m_{i-1})^{4(c+1)^{2^i}} \forall i \geq 1$ $(c \geq 1)$.

Let $c_1 = c+1$.

▶ **Claim 20.** $m_i \leq (3c_m)^{(4c_1)^{2^{i+1}}}$.

Proof. For $i = 0$, the right hand side evaluates to $(3c_m)^{(4c_1)^2} \geq c_m$, which is equal to the left hand side. Now we assume that the statement is true for $i-1$ and prove for i where $i \geq 1$. $m_i \leq (3m_{i-1})^{4c_1^{2^i}} \leq (3(3c_m)^{(4c_1)^{2^i}})^{4c_1^{2^i}} = 3^{(4c_1)^{2^i}4c_1^{2^i}+4c_1^{2^i}} c_m^{(4c_1)^{2^i}4c_1^{2^i}} \leq (3c_m)^{(4c_1)^{2^{i+1}}}$ as $4^{2^i+1} + 4 \leq 4^{2^{i+1}} \forall i \geq 1$.

We have $v_i \leq m_i$. Thus, the size of graph A_i is at most $(3c_m)^{(4c_1)^{2^{i+1}}}$. ◀

◀

G Proof of Theorem 15

Proof. Think of G_1 and G_2 as undirected G_1' and G_2'; their sparsity remains the same. Let H be the set of edges on the cut that achieves the sparsest cut on G separating n source-sink pairs. Consider partitioning this set into sets $H_i = \{e_{1i}, e_{2i}, ..., e_{h_i i}\}$ according to which copy of G_2 (or equivalently G_2'), the edge belongs to in G. H_i denotes the edges belonging to the i-th copy of G_2, $|H_i| = h_i$. Note that $|H| = \sum_i h_i$. Let $n_i^{(2)}$ be the number of source and sink pairs that H_i separates in the i-th copy of G_2. These cuts have capacity $\sum_{e\in H_i} c_{2e}$ in G_2. By construction, each of these source-sink pairs would have replaced an edge in some copy of G_1 (or equivalently undirected G_1'). Assume the k-th ($k \in [n_i^{(2)}]$) source-sink pair replaced edge e_i in the j_{ik}-th copy of G_1 (All source-sink pairs replace the same edge). Mark this edge in the j_{ik}-th copy of G_1 (which has now been replaced in G). The i-th copy of G_2 makes $n_i^{(2)}$ marks. Let F_j be the set of all such marked edges in the j-th copy of G_1. Let F_j cut $n_j^{(1)}$ source-sink pairs in G_1. Any source-sink pair that gets cut in G by H must be cut in G_1 under F_j by construction. Therefore, $\sum_j n_j^{(1)} \geq n$. It is not an equality because there could be a source-sink pair that gets cut by F_j but not by H in G, due to paths that travel from the source to other copies of G_1 through connecting copies of G_2 and come back at the sink. The theorem follows from the following inequalities:

$$\begin{aligned}
\sum_{e\in H} c_e = \sum_i c_{1e_i} \sum_{e\in H_i} c_{2e} &= \sum_i n_i^{(2)} c_{1e_i} \frac{\sum_{e\in H_i} c_{2e}}{n_i^{(2)}} \\
&\geq \sum_i n_i^{(2)} c_{1e_i} \text{Sparsity}(G_2) = \text{Sparsity}(G_2)\left(\sum_i n_i^{(2)} c_{1e_i}\right) \\
&= \text{Sparsity}(G_2)\left(\sum_j \sum_{e\in F_j} c_{1e}\right) = \text{Sparsity}(G_2)\left(\sum_j n_j^{(1)} \frac{\sum_{e\in F_j} c_{1e}}{n_j^{(1)}}\right) \\
&\geq \text{Sparsity}(G_2)\left(\sum_j n_j^{(1)}\right) \text{Sparsity}(G_1) \geq n\left(\text{Sparsity}(G_1)\cdot\text{Sparsity}(G_2)\right)
\end{aligned} \tag{6}$$

The first equality follows from the definition of edge capacities in G in terms of edge capacities in G_1 and G_2. Since H_i cuts $n_i^{(2)}$ source-sink pairs in a copy of G_2, the first inequality follows from the Sparsity(G_2) being the smallest ratio for all the cuts. The first equality on the third line follows from the fact that an edge belongs to F_j only when the corresponding source-sink pair that replaced this edge in G_1 is cut by the cut corresponding to that copy of G_2 and i-th copy of G_2 result in exactly $n_i^{(2)}$ such edges distributed amongst F_js. Therefore, $\frac{\sum_{e\in H} c_e}{n} \geq \text{Sparsity}(G_1)\cdot\text{Sparsity}(G_2) \implies \text{Sparsity}(G) \geq \text{Sparsity}(G_1)\cdot \text{Sparsity}(G_2)$. Here, we assumed that all the demands are 1 in graphs G_1, G_2 and G. ◀