Network Coding Gaps for Completion Times of Multiple Unicasts*

Bernhard Haeupler, David Wajc, and Goran Zuzic

Carnegie Mellon University
{haeupler, dwajc, gzuzic}@cs.cmu.edu

Abstract

We study network coding gaps for the problem of makespan minimization of multiple unicasts. In this problem distinct packets at different nodes in a network need to be delivered to a destination specific to each packet, as fast as possible. The *network coding gap* specifies how much coding packets together in a network can help compared to the more natural approach of routing.

While makespan minimization using routing has been intensely studied for the multiple unicasts problem, no bounds on network coding gaps for this problem are known. We develop new techniques which allow us to upper bound the network coding gap for the makespan of k unicasts, proving this gap is at most polylogarithmic in k. Complementing this result, we show there exist instances of k unicasts for which this coding gap is polylogarithmic in k. Our results also hold for average completion time, and more generally any ℓ_p norm of completion times.

*Supported in part by NSF grants CCF-1618280, CCF-1814603, CCF-1527110, NSF CAREER award CCF-1750808, and a Sloan Research Fellowship, as well as a DFINITY scholarship.

1 Introduction

In this paper we study the natural mathematical abstraction of what is arguably the most common network communication problem: *multiple unicasts*. In this problem, distinct packets of different size are at different nodes in a network, and each packet needs to be delivered to a specific destination as fast as possible. That is, minimizing the *makespan*, or the time until all packets are delivered.

All known multiple-unicast solutions employ (fractional) *routing* (also known as store-and-forward protocols), i.e., network nodes potentially subdivide packets and route (sub-)packets to their destination via store and forward operations, while limited by edge capacities. The problem of makespan minimization of routing has been widely studied over the years. A long line of work [9, 11, 14, 44, 49, 50, 57, 58, 59, 63, 65, 66, 67], starting with the seminal work of Leighton, Maggs, and Rao [49], studies makespan minimization for routing along fixed paths. The study of makespan minimization for routing (with the freedom to pick paths along which to route) resulted in approximately-optimal routing, first for asymptotically-large packet sizes [11], and then for all packet sizes [67].

It seems obvious at first that routing packets, as though they were physical commodities, is the only way to solve network communication problems, such as multiple unicasts. Surprisingly, however, results discovered in the 2000s [6] suggest that information need not flow through a network like a physical commodity. For example, nodes might not just forward information, but instead send out XORs of received packets. Multiple such XORs or linear combinations can then be recombined at destinations to reconstruct any desired packets. An instructive example is to look at the XOR $C \oplus M$ of two s-bit packets, C and M. While it is also s bits long, one can use it to reconstruct either all s bits of C or all s bits of M, as long as the other packet is given. Such network coding operations are tremendously useful for network communication problems, but they do not have a physical equivalent. Indeed, the $C \oplus M$ packet would correspond to some s ounces of a magic "café latte" liquid with the property that one can extract either s ounces of milk or s ounces of coffee from it, as long as one has enough of the other liquid already. Over the last two decades, many results demonstrating gaps between the power of network coding and routing have been published (e.g., [6, 7, 16, 20, 24, 27, 29, 30, 36, 42, 47, 52, 68, 69, 70]). Attempts to build a comprehensive theory explaining what is or is not achievable by going beyond routing have given rise to an entire research area called network information theory.

The question asked in this paper is:

"How much faster than routing can network coding be for any multiple-unicast instance?"

In other words, what is the (multiplicative) *network coding gap* for makespan of multiple unicasts. Surprisingly, no general makespan coding gap bounds were known prior to this work. This is in spite of the vast amount of effort invested in understanding routing strategies for this problem, and ample evidence of the benefits of network coding.

This question was studied in depth for the special case of *asymptotically-large* packet sizes, otherwise known as *throughput* maximization (e.g., [3, 7, 36, 37, 41, 45, 47, 52, 52, 70, 71]). Here, the maximum *throughput* of a multiple-unicast instance can be defined as $\sup_{w \to \infty} w/C(w)$, where $C(w)$ is the makespan of the fastest protocol for the instance after increasing all packet sizes by a factor of w (see Appendix A). In the throughput setting, no instances are known where coding offers *any* advantage over routing, and this is famously conjectured to be the case for all instances [37, 52]. This conjecture, if true, has been proven to have surprising connections to various lower bounds [3, 4, 23]. Moreover, by the work Afshani et al. [4], a throughput coding gap of $o(\log k)$ for all multiple-unicast instances with k unicast pairs (k-*unicast instances*, for short) would imply explicit *super-linear circuit lower bounds*—a major breakthrough in complexity theory. Such a result is currently out of reach, as the best known upper bound on throughput coding gaps is $O(\log k)$, which follows easily from the same bound on multicommodity flow/sparsest cut gaps [10, 54].

In this work we prove makespan coding gaps for the general problem of *arbitrary* packet sizes. In particular, we show that this gap is at most $O(\log^2 k)$ for any k-unicast instance (for the most interesting case of similar-sized packet sizes). We note that any coding gap upper bound for this more general setting immediately implies the same bound in the throughput setting (Appendix A), making our general bound only quadratically larger than the best known bound for the special case of throughput. Complementing our results, we prove that there exist k-unicast instances where the network coding gap is $\Omega(\log^c k)$ for some constant $c > 0$.

To achieve our results we develop novel techniques that might be of independent interest. The need for such new tools is due to makespan minimization for general packet sizes needing to take both source-sink distances as well as congestion issues into account. This is in contrast with the throughput setting, where bounds must only account for congestion, since asymptotically-large packet sizes make distance considerations inconsequential. For our more general problem, we must therefore develop approaches that are both congestion- and distance-aware. One such approach is given by a novel combinatorial object we introduce, dubbed the *moving cut*, which allows us to provide a *universally optimal* characterization of the coding makespan. That is, it allows us to obtain tight bounds (up to polylog terms) on the makespan of *any* given multiple-unicast instance. This underlies our main result—a polylogarithmic upper bound on the makespan coding gap for any multiple-unicast instance.

1.1 Preliminaries

In this section we define the completion-time communication model. We defer the, slightly more general, information-theoretic formalization to Appendix B.

A *multiple-unicast instance* $\mathcal{M} = (G, \mathcal{S})$ is defined over a communication network, represented by an undirected graph $G = (V, E, c)$ with capacity $c_e \in \mathbb{Z}_{\geq 1}$ for each edge e. The $k \triangleq |\mathcal{S}|$ *sessions* of $\mathcal{M}$ are denoted by $\mathcal{S} = \{(s_i, t_i, d_i)\}_{i=1}^k$. Each session consists of source node s_i, which wants to transmit a packet to its sink t_i, consisting of $d_i \in \mathbb{Z}_{\geq 1}$ sub-packets; we refer to d_i as the *demand*. Without loss of generality we assume that a uniform sub-packetization is used; i.e., all sub-packets have the same size (think of sub-packets as the underlying data type, e.g., field elements or bits; our proofs hold for any particular sub-packet). For brevity, we refer to an instance with k sessions as a *k-unicast* instance.

A *protocol* for a multiple-unicast instance is conducted over finitely-many *synchronous time steps*. Initially, each source s_i knows its packet, consisting of d_i sub-packets. At any time step, the protocol instructs each node v to send a different packet along each of its edges e. The packet contents are computed with some predetermined function of packets received in prior rounds by v or originating at v. *Network coding protocols* are unrestricted protocols, allowing each node to send out any function of the packets it has received so far. On the other hand, *routing protocols* are a restricted, only allowing a node to forward sub-packets which it has received so far or that originate at this node.

We say a protocol for multiple-unicast instance has *completion times* $(T_1, T_2, \ldots, T_k)$ if for each $i \in [k]$, after T_i time steps of the protocol the sink t_i can determine the d_i-sized packet of its source s_i. The complexity of a protocol is determined by functions $\mathcal{C} : \mathbb{R}^k_{\geq 0} \to \mathbb{R}_{\geq 0}$ of its completion times. For example, a protocol with completion times $(T_1, T_2, \ldots, T_k)$ has *makespan* $\max_{i\in[k]} T_i$ and *average completion time* $(\sum_{i\in[k]} T_i)/k$. Minimizing these measures is a special case of minimizing weighted ℓ_p norms of completion time, namely minimizing $(\sum_{i\in[k]} w_i \cdot T_i^p)^{1/p}$ for some $\vec{w} \in \mathbb{R}^k$ and $p \in \mathbb{R}_{\geq 0}$.

Since coding protocols subsume routing ones, for any function $\mathcal{C}$ of completion times, and for any multiple-unicast instance, the fastest routing protocol is no faster than the fastest coding protocol. Completion-time coding gaps characterize how much faster the latter is.

Definition 1.1. *(Completion-time coding gaps) For any function $\mathcal{C} : \mathbb{R}^k_{\geq 0} \to \mathbb{R}_{\geq 0}$ of completion times, the* network coding gap *for $\mathcal{C}$ for a k-unicast instance $\mathcal{M} = (G, \mathcal{S})$ is the ratio of the smallest $\mathcal{C}$-value of any routing protocol for $\mathcal{M}$ and the smallest $\mathcal{C}$-value of any network coding protocol for $\mathcal{M}$.*

We note that the multiple-unicast instance problem can be further generalized, so that each edge has both capacity and *delay*, corresponding to the amount of time needed to traverse the edge. This more general problem can be captured by replacing each edge e with a path with unit delays of total length proportional to e's delay. As we show, despite path length being crucially important in characterizing completion times for multiple-unicast instances, this transformation does not affect the worst-case coding gaps, which are independent of the network size (including after this transformation). We therefore consider only unit-time delays in this paper, without loss of generality.

1.2 Our Contributions

In this work we show that completion-time coding gaps of multiple unicasts are vastly different from their throughput counterparts, which are conjectured to be trivial (i.e., equal to one). For example, while the throughput coding gap is always one for instances with $k = 2$ sessions [39], for makespan it is easy to derive instances with $k = 2$ sessions and coding gap of $4/3$ (based on the butterfly network). Having observed that makespan coding gaps can in fact be nontrivial, we proceed to study the potential asymptotic growth of such coding gaps as the network parameters grow. We show that the makespan coding gap of multiple unicasts with k sessions and packet sizes $\{d_i\}_{i\in[k]}$ is polylogarithmic in the problem parameters, k and $\sum_i d_i/\min_i d_i$, but independent of the network size, n. The positive part of this result is given by the following theorem.

Theorem 1.2. *The network coding gap for makespan of any k-unicast instance is at most*

$$O\left(\log(k)\cdot\log\left(\sum_i d_i/\min_i d_i\right)\right).$$

For similarly-sized packets, this bound simplifies to $O(\log^2 k)$. For different-sized packets, our proofs and ideas in [62] imply a coding gap of $O(\log k\cdot\log(nk))$. Moreover, our proofs are constructive, yielding for any k-unicast instance $\mathcal{M}$ a routing protocol which is at most $O(\log k\cdot\log(\sum_i d_i/\min_i d_i))$ and $O(\log k\cdot\log(nk))$ times slower than the fastest protocol (of any kind) for $\mathcal{M}$. We note that our upper bounds imply the same upper bounds for throughput (see Appendix A). Our bounds thus also nearly match the best coding gap of $O(\log k)$ known for this special case of makespan minimization.

On the other hand, we prove that a polylogarithmic gap as in Theorem 1.2 is inherent, by providing an infinite family of multiple-unicast instances with unit-sized packets ($d_i = 1$ for all $i\in[k]$) exhibiting a polylogarithmic makespan coding gap.

Theorem 1.3. *There exists an absolute constant $c > 0$ and an infinite family of k-unicast instances whose makespan coding gap is at least $\Omega(\log^c k)$.*

Building on our results for makespan we obtain similar results to Theorems 1.2 and 1.3 for average completion time and more generally for any weighted ℓ_p norm of completion times.

1.3 Techniques

Here we outline the challenges faced and key ideas needed to obtain our results, focusing on makespan.

1.3.1 Upper Bounding the Coding Gap

As we wish to bound the ratio between the best makespan of any routing protocol and any coding protocol, we need both upper and lower bounds for these best makespans. As it turns out, upper bounding the best makespan is somewhat easier. The major technical challenge, and our main contribution, is in deriving lower bounds on the optimal makespan of any given multiple-unicast instance. Most notably, we formalize a technique we refer to as the *moving cut*. Essentially the same technique was used to prove that distributed verification is hard on one *particular* graph that was designed specifically with this technique in mind [19, 21, 61]. Strikingly, we show that the moving cut technique gives an almost-tight characterization (up to polylog factors) of the coding makespan for *every* multiple-unicast instance (i.e., it gives *universally* optimal bounds).

We start by considering several prospective techniques to prove that no protocol can solve an instance in fewer than T rounds, and build our way up to the moving cut. For any multiple-unicast instance, $\max_{i\in[k]} \text{dist}(s_i, t_i)$, the maximum distance between any source-sink pair, clearly lower bounds the coding makespan. However, this lower bound can be arbitrarily bad since it does not take edge congestion into account; for example, if all source-sink paths pass through one common edge. Similarly, any approach that looks at sparsest cuts in a graph is also bound to fail since it does not take the source-sink distances into account.

Attempting to interpolate between both bounds, one can try to extend this idea by noting that a graph that is "close" (in the sense of few deleted edges) to another graph with large source-sink distances must have large makespan for routing protocols. For simplicity, we focus on instances where all capacities and demands are one, i.e., $c_e = 1$ for every edge e and $d_i = 1$ for all i, which we refer to as *simple* instances. The following simple lemma illustrates such an approach.

Lemma 1.4. *Let $\mathcal{M} = (G, \mathcal{S})$ be a simple k-unicast instance. Suppose that after deleting some edges $F \subseteq E$, any sink is at distance at least T from its source; i.e., $\forall i \in [k]$ we have $dist_{G\setminus F}(s_i, t_i) \geq T$. Then any routing protocol for $\mathcal{M}$ has makespan at least* $\min\{T, k/|F|\}$.

Proof. For any sets of flow paths between all sinks and source, either (1) all source-sink flow paths contain at least one edge from F, incurring a congestion of $k/|F|$ on at least one of these $|F|$ edges, or (2) there is a path not containing any edge from F, hence having a hop-length of at least T. Either way, any routing protocol must take at least $\min\{T, k/|F|\}$ to route along these paths. □

Perhaps surprisingly, the above bound does not apply to general (i.e., coding) protocols. Consider the instance in Figure 1. There, removing the single edge $\{S, T\}$ increases the distance between any source-sink pair to 5, implying any routing protocol's makespan is at least 5 on this instance. However, there exists a network coding protocol with makespan 3: Each source s_i sends its input to its neighbor S and all sinks t_j for $i \neq j$ along the direct 3-hop path $s_i - - - t_j$. Node S computes the XOR of all inputs, passes this XOR to T who, in turn, passes this XOR to all sinks t_j, allowing each sink t_j to recover its source s_j's packets by canceling all other terms in the XOR.

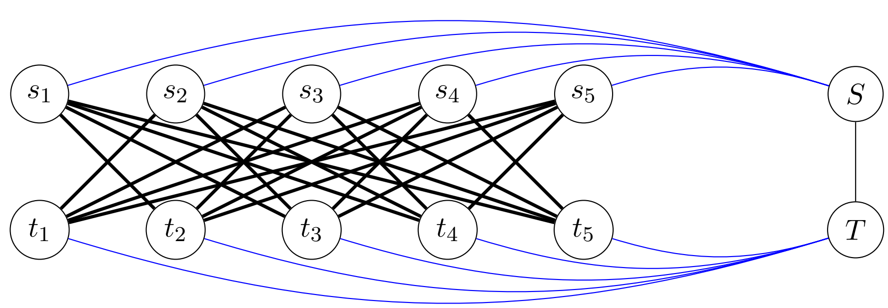

Figure 1: A family of instances with $k = 5$ pairs of terminals and makespan coding gap of $5/3$. Each of the k sources s_i has a path of 3 hops (in black and bold) connecting it to every sink t_j for all $j \neq i$. Moreover, all sources s_i neighbor a node S, which also neighbors node T, which neighbors all sinks t_j.

One can still recover a valid general (i.e., coding) lower bound by an appropriate strengthening of Lemma 1.4: one has to require that *all* sources be far from *all* sinks in the edge-deleted graph. This contrast serves as a good mental model for the differences between coding and routing protocols.

Lemma 1.5. *Let $\mathcal{M} = (G, \mathcal{S})$ be a simple k-unicast instance. Suppose that after deleting some edges $F \subseteq E$, any sink is at distance at least T from* **any** *source; i.e., $\forall \boldsymbol{i}, \boldsymbol{j} \in [k]$ we have $dist_{G \setminus F}(\boldsymbol{s_i}, \boldsymbol{t_j}) \geq T$. Then any (network coding) protocol for $\mathcal{M}$ has makespan at least* $\min\{T, k/|F|\}$.

Proof. We can assume all sources can share information among themselves for free (e.g., via a common controlling entity) since this makes the multiple-unicast instance strictly easier to solve; similarly, suppose that the sinks can also share information. Suppose that some coding protocol has makespan $T' < T$. Then all information shared between the sources and the sinks has to pass through some edge in F at some point during the protocol. However, these edges can pass a total of $|F| \cdot T'$ packets of information, which has to be sufficient for the total of k source packets. Therefore, $|F| \cdot T' \geq k$, which can be rewritten as $T' \geq k/|F|$. The makespan is therefore at least $T' \geq \min\{T, k/|F|\}$. □

Unfortunately, Lemma 1.5 is not always tight and it is instructive to understand when this happens. One key example is the previously-mentioned instance studied in the influential distributed computing papers [19, 21, 61] (described in Figure 2), where congestion and dilation both play key roles. Informally, this network was constructed precisely to give an $\tilde{\Omega}(\sqrt{n})$ makespan lower bound (leading to the pervasive $\tilde{\Omega}(\sqrt{n} + D)$ lower bound for many global problems in distributed computing [19]). The intuitive way to explain the $\tilde{\Omega}(\sqrt{n})$ lower bound is to say that one either has to communicate along a path of length $\sqrt{n}$ or *all* information needs to shortcut significant distance over the tree, which forces all information to pass through near the top of the tree, implying congestion of $\tilde{\Omega}(\sqrt{n})$. Lemma 1.5, however, can at best certify a lower bound of $\tilde{\Omega}(n^{1/4})$ for this instance. That is, this lemma's (coding) makespan lower bound can be *polynomially* far from the optimal coding protocol's makespan.

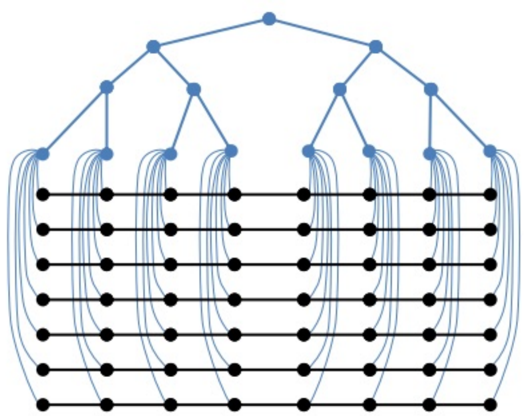

Figure 2: The hard instance for distributed graph problems [19, 21, 61], as appears in [25]. The multiple-unicast instance has $\Theta(n)$ nodes and is composed of $\sqrt{n}$ disjoint paths of length $\sqrt{n}$ and a perfectly balanced binary tree with $\sqrt{n}$ leaves. The i^{th} node on every path is connected to the i^{th} leaf in the tree. There are $\sqrt{n}$ sessions with s_i, t_i being the first and last node on the i^{th} path. All capacities and demands are one. The graph's diameter is $\Theta(\log n)$, but its coding makespan is $\tilde{\Omega}(\sqrt{n})$.

A more sophisticated argument is needed to certify the $\tilde{\Omega}(\sqrt{n})$ lower bound for this specific instance. The aforementioned papers [19, 21, 61] prove their results by implicitly using the technique we formalize as our moving cut in the following lemma (proven in Section 2.1) on the instance of Figure 2.

Lemma 1.6 (Moving cut)**.** *Let $\mathcal{M} = (G, \mathcal{S})$ be a k-unicast instance. Suppose that after increasing each edge e's length from one to $\ell_e \in \mathbb{Z}_{\geq 1}$, we have*

(1) the total length increase, $\sum_{e \in E} c_e(\ell_e - 1)$ is less than $\sum_{i=1}^{k} d_i$, and

(2) any sink is at distance at least T from any source; i.e., $\forall i, j \in [k]$ we have $dist_\ell(s_i, t_j) \geq T$.

Then any *(coding) protocol for $\mathcal{M}$ has makespan at least T.*

Lemma 1.6 can be seen as a natural generalization of Lemma 1.5, which can be equivalently restated in the following way: *"Suppose that after increasing each edge e's length from one to* $\ell_e \in \{\mathbf{1}, \boldsymbol{T+1}\}$*, we have that (1)* $\sum_{e\in E} c_e(\ell_e - 1) < \sum_{i=1}^{k} d_i$*, and (2)* $dist_\ell(s_i, t_j) \geq T$*. Then any (coding) protocol* $\mathcal{M}$ *has makespan at least* T*"*. Dropping the restriction on ℓ_e recovers Lemma 1.6.

Strikingly, the moving cut technique allows us not only to prove tight bounds (up to polylog factors) for the instance of Figure 2—it allows us to get such tight bounds *for every multiple-unicast instance*. In order to upper bound the makespan coding gap, we therefore relate such a moving cut with the optimal routing makespan, as follows.

To characterize the optimal routing makespan, we study hop-bounded multicommodity flow, which is an LP relaxation of routing protocols of makespan T. First, we show that a fractional LP solution of high value to this LP implies a routing with makespan $O(T)$. Conversely, if the optimal value of this LP is low, then by strong LP duality this LP's dual has a low-valued solution, which we us to derive a moving cut and lower bound the coding makespan. Unfortunately, the dual LP only gives us bounds on (average) distance between source-sink pairs (s_i, t_i), and not between all sources s_i and sinks t_j (including $j \neq i$), as needed for moving cuts. For this conversion to work, we prove a generalization of the main theorem of Arora, Rao and Vazirani [8] to general metrics, of possible independent interest. (See Section 2.3.) This allows us to show that a low-valued dual solution implies a moving cut certifying that no coding protocol has makespan less than $T/O(\log k \cdot \log(\sum_i d_i / \min_i d_i))$. As the above rules out low-valued optimal solutions to the LP for $T = T^* \cdot O(\log k \cdot \log(\sum_i d_i / \min_i d_i))$ with T^* the optimal coding makespan, the LP must have high optimal value, implying a routing protocol with makespan $O(T)$, and thus our claimed upper bound on the makespan coding gap.

1.3.2 Lower Bounding the Coding Gap

To complement our polylogarithmic upper bound on the makespan gap, we construct a family of multiple-unicast instances $\mathcal{M}$ that exhibit a polylogarithmic makespan coding gap. We achieve this by amplifying the gap via graph products, a powerful technique that was also used in prior work to construct extremal throughput network coding examples [12, 13, 55]. Here we outline this approach, as well as the additional challenges faced when trying to use this approach for makespan.

We use a graph product introduced by Braverman et al. [13] (with some crucial modifications). Braverman et al. [13] use their graph product to prove a conditional *throughput* coding gap similar to the one of Theorem 1.3, conditioned on the (unknown) existence of a multiple-unicast instance I with non-trivial throughput coding gap. The graph product of [13] takes instances I_1, I_2 and intuitively replaces each edge of I_1 with a source-sink pair of a different copy of I_2. More precisely, multiple copies of I_1 and I_2 are created and interconnected. Edges of a copy of I_1 are replaced by the same session of different copies of I_2; similarly, sessions of a copy of I_2 replace the same edge in different copies of I_1. This product allows for coding protocols in I_1 and I_2 to compose in a straightforward way to form a fast coding protocol in the product instance. The challenge is in proving impossibility results for routing protocols, which requires more care in the definition of the product graph.

To address this challenge, copies of instances are interconnected along a high-girth bipartite graph to prevent unexpectedly short paths from forming after the interconnection. For example, to prove a throughput routing impossibility result, Braverman et al. [13] compute a dual of the multicommodity flow LP (analogous to our LP, but *without any hop restriction*) to certify a limit on the routing performance. In the throughput setting, a direct tensoring of dual LP solutions of I_1 and I_2 gives a satisfactory dual solution of the product instance. In more detail, a dual LP solution in I assigns a positive length $\ell_I(e)$ to each edge in I; each edge of the product instance corresponds to two edges $e_1 \in I_1$ and $e_2 \in I_2$, and the direct tensoring $\ell_+((e_1, e_2)) = \ell_{I_1}(e_1) \cdot \ell_{I_2}(e_2)$ provides a feasible dual solution with an adequate objective value. To avoid creating edges in the product distance of zero ℓ_+-length, they contract edges assigned length zero in the dual LP of either instance. Unfortunately for us, such contraction is out of the question when studying makespan gaps, as such contractions would shorten the hop length of paths, possibly creating short paths with no analogues in the original instance.

Worse yet, any approach that uses the dual of our T-hop-bounded LP is bound to fail in the makespan setting. To see why, suppose we are given two instances I_1, I_2, both of which have routing makespan at least T and expect that the product instance I_+ to have routing makespan at least T^2 by some construction of a feasible dual LP solution. Such a claim cannot be directly argued since a source-sink path in the product instance that traverses, say, $T-1$ different copies of I_2 along a path of hop-length $T+1$ in I_2 could carry an arbitrary large capacity! This is since the hop-bounded LP solution on I_2 only takes *short* paths, of hop-length at most T, into account. Since there is no direct way to compose the dual LP solutions, we are forced to use a different style of analysis from the one of [13], which in turn forces our construction to become considerably more complicated.

To bound the routing *makespan* in the product instance we rely on Lemma 1.4: We keep a list of edges F along with each instance and ensure that (i) all source-sink distances in the F-deleted instance are large and that (ii) the ratio of the number of sessions k to $|F|$ is large. We achieve property (i) by interconnecting along a high-girth graph and treating the replacements of edges in F in a special way (hence deviating from the construction of [13]). Property (ii) is ensured by making the inner instance I_2 significantly larger than the outer instance I_1, thus requiring many copies of I_1 and resulting in a large number of sessions in the product graph. To allow for this asymmetric graph product, we need an infinite number of base cases with non-trivial makespan coding gap for our recursive constructions (rather than a single base instance, as in the work of Braverman et al. [13]). This infinite family is fortunately obtained by appropriately generalizing the instance of Figure 1.

The main challenge in our approach becomes controlling the size of the product instance. To achieve this, we affix to each instance a relatively complicated set of parameters (e.g., coding makespan, number of edges, number of sessions, etc.) and study how these parameters change upon applying the graph product. Choosing the right set of parameters is key—they allow us to properly quantify the size escalation. In particular, we show that the coding gap grows doubly-exponentially and the size of the instances grow triply-exponentially, yielding the desired polylogarithmic coding gap.

1.4 Related Work

This work ties in to many widely-studied questions. We outline some of the most relevant here.

Routing multiple unicasts. Minimizing the makespan of multiple unicasts using routing has been widely studied. When packets must be routed along fixed paths, two immediate lower bounds on the makespan emerge: *dilation*, the maximum length of a path, and *congestion*, the maximum number of paths crossing any single edge. A seminal result of Leighton et al. [49] proves one can route along such fixed paths in $O(\text{congestion} + \text{dilation})$ rounds, making the result optimal up to constants. Follow ups include works improving the constants in the above bound [59, 66], computing such protocols [50], simplifying the original proof [65], routing in distributed models [57, 63], and so on. When one has the freedom to choose paths, Bertsimas and Gamarnik [11] gave near-optimal routing solutions for asymptotically-large packet sizes, later extended to all packet sizes by Srinivasan and Teo [67]. The power of routing for multiple unicasts is therefore by now well understood.

Network coding gains. The utility of network coding became apparent after Ahlswede et al. [6] proved it can increase the (single multicast) throughput of a communication network. Following their seminal work, there emerged a vast literature displaying the advantages of network coding over routing for various measures of efficiency in numerous communication models, including for example energy usage in wireless networks [24, 27, 69], delay minimization in repeated single unicast[16, 68], and makespan in gossip protocols [20, 29, 30]. The throughput of a single multicast (i.e., one single node sending to some set of nodes), arguably the simplest non-trivial communication task, was also studied in great detail (e.g., [5, 6, 38, 40, 51, 53]). In particular, Agarwal and Charikar [5] showed that the throughput coding gap for a single multicast equals the integrality gap of natural min-weight Steiner tree relaxations, for which non-trivial bounds were known (see, e.g., [34, 72]). While the throughput coding gap for a single multicast is now fairly well understood, the case of multiple senders seems to be beyond the reach of current approaches.

Throughput gaps for multiple unicasts. The routing throughput for multiple unicasts is captured by multicommodity max-flow, while the coding throughput is clearly upper bounded by the sparsest cut. Known multicommodity flow-cut gap bounds therefore imply the throughput coding gap for k unicasts is at most $O(\log k)$ [10, 54], and less for special families of instances [15, 17, 18, 43, 46, 48, 64]. In 2004 Li and Li [52] and Harvey et al. [37] independently put forward the *multiple-unicast conjecture*, which asserts that the throughput coding gap is trivial (i.e., it is one). This conjecture was proven true for numerous classes of instances [3, 39, 41, 45, 56]. More interestingly, a positive resolution of this conjecture has been shown to imply unconditional lower bounds in external memory algorithm complexity [3, 23], computation in the cell-probe model [3], and (recently) an $\Omega(n \log n)$ circuit size lower bound for multiplication of n-bit integers [4] (matching an even more recent breakthrough algorithmic result for this fundamental problem [35]). Given this last implication, it is perhaps not surprising that despite attempts by many prominent researchers [7, 36, 41, 47, 52, 70, 71], the conjecture remains open and has established itself as a notoriously hard open problem. Indeed, even improving the $O(\log k)$ upper bound on throughput coding gaps seems challenging, and would imply unconditional *super-linear circuit size lower bounds*, by the work of Afshani et al. [4]. Improving our upper bound on makespan coding gaps to $o(\log k)$ would directly imply a similar improvement for throughput coding gaps, together with these far-reaching implications.

2 Upper Bounding the Coding Gap

In this section we prove Theorem 1.2, upper bounding the makespan network coding gap. Given a multiple-unicast instance $\mathcal{M}$ we thus want to upper bound its routing makespan and lower bound its coding makespan. To characterize these quantities we start with a natural hop-bounded multicommodity flow LP, $\textsc{ConcurrentFlow}_{\mathcal{M}}(T)$, which serves as a "relaxation" of routing protocols of makespan at most T. The LP, given in Figure 3, requires to send a flow of magnitude $z \cdot d_i$ between each source-sink pair (s_i, t_i), with the additional constraints that (1) the combined congestion of any edge e is at most $T \cdot c_e$ where c_e is the capacity of the edge (as only c_e packets can use this edge during any of the T time steps of a routing protocol), and (2) the flow is composed of only *short* paths, of at most T hops. Specifically, for each $i \in [k]$, we only route flow from s_i to t_i along paths in $\mathcal{P}_i(T) \triangleq \{p : s_i \rightsquigarrow t_i \mid |p| \le T,\ p \text{ is simple}\}$, the set of simple paths of hop-length at most T connecting s_i and t_i in G.

Primal: $\textsc{ConcurrentFlow}_{\mathcal{M}}(T)$		Dual: $\textsc{Cut}_{\mathcal{M}}(T)$	
maximize	z	minimize	$T \cdot \sum_{e \in E} c_e \ell_e$
subject to:		subject to:	
$\forall i \in [k]$:	$\sum_{p \in \mathcal{P}_i(T)} f_i(p) \ge z \cdot d_i$	$\forall i \in [k], p \in \mathcal{P}_i(T)$:	$\sum_{e \in p} \ell_e \ge h_i$
$\forall e \in E$:	$\sum_{p \ni e} f_i(p) \le T \cdot c_e$		$\sum_{i \in [k]} d_i h_i \ge 1$
$\forall i \in [k], p$:	$f_i(p) \ge 0$	$\forall e \in E$:	$\ell_e \ge 0$
		$\forall i \in [k]$:	$h_i \ge 0$

Figure 3: The concurrent flow LP relaxation and its dual.

A routing protocol solving $\mathcal{M}$ in T rounds yields a solution to $\textsc{ConcurrentFlow}_{\mathcal{M}}(T)$ of value $z = 1$, almost by definition.[1] A partial converse is also true; a feasible solution to $\textsc{ConcurrentFlow}_{\mathcal{M}}(T)$ of value at least $\Omega(1)$ implies a routing protocol for $\mathcal{M}$ in time $O(T)$. This can be proven using standard LP rounding [67] and $O(\text{congestion} + \text{dilation})$ path routing [49]. (See Appendix C.)

Lemma 2.1. *Let $z \leq 1, \{f_i(p) \mid i \in [k], p \in \mathcal{P}_i(T)\}$ be a feasible solution for $\textsc{ConcurrentFlow}_{\mathcal{M}}(T)$. Then there exists an integral routing protocol with makespan $O(T/z)$.*

Surprisingly (and more challenging to prove), a low optimal LP value for $\textsc{ConcurrentFlow}_{\mathcal{M}}(T)$ implies that no *coding* protocol can solve the instance in much less than T time.

Lemma 2.2. *If the optimal value of $\textsc{ConcurrentFlow}_{\mathcal{M}}(T)$ is at most $z^* \leq 1/10$, then the coding makespan for $\mathcal{M}$ is at least $T/(C \cdot \log(k) \cdot \log(\sum_i d_i/d_i))$ for some constant $C > 0$.*

Before outlining our approach for proving Lemma 2.2, we show why this lemma together with Lemma 2.1 implies our claimed upper bound for the makespan network coding gap.

Theorem 1.2. *The network coding gap for makespan of any k-unicast instance is at most*

$$O\left(\log(k) \cdot \log\left(\sum_i d_i / \min_i d_i\right)\right).$$

Proof. Fix a multiple-unicast instance $\mathcal{M}$. Let T^* be the minimum makespan for any coding protocol for $\mathcal{M}$. Let $T = (C+1) \cdot T^* \cdot (\log(k) \cdot \log(\sum_i d_i / \min_i d_i))$ for C as in Lemma 2.2. Then, the LP $\textsc{ConcurrentFlow}_{\mathcal{M}}(T)$ must have optimal value at least $z^* \geq 1/10$, else by Lemma 2.2 and our choice of T, any coding protocol has makespan at least $T/O(\log(k) \cdot \log(\sum_i d_i / \min_i d_i)) > T^*$, contradicting the definition of T^*. But then, by Lemma 2.1, there exists a routing protocol with makespan $O(T/z^*) = O(T^* \cdot \log(k) \cdot \log(\sum_i d_i / \min_i d_i)$. The theorem follows. □

The remainder of the section is dedicated to proving Lemma 2.2. That is, proving that a low optimal value for the LP $\textsc{ConcurrentFlow}_{\mathcal{M}}(T)$ implies a lower bound on the makespan of any coding protocol for $\mathcal{M}$. To this end, we take a low-valued LP solution to the dual LP $\textsc{Cut}_{\mathcal{M}}(T)$ (implied by strong LP duality) and use it to obtain an information-theoretic certificate of impossibility, which we refer to as a *moving cut*. Section 2.1 introduces the framework to prove such certificates of impossibility and Section 2.2 explains the transformation of a low-value dual LP solution to such a moving cut. For this transformation, we prove a lemma reminiscent of Arora et al. [8, Theorem 1] for general metrics (see Section 2.3).

2.1 Moving Cut

In this section we present our makespan lower bound certificates, which we term the *moving cut*. Informally, given an underlying multiple-unicast instance, a moving cut is a sequence of nested cuts (represented as vertex sets) each containing all of the sources, i.e., $\{s_i\}_i \subseteq B_0 \subseteq B_1 \subseteq B_2 \subseteq \ldots$, and where none of the first T sets contain any of the sinks, i.e., $B_t \cap \{t_i\}_i = \emptyset$. A good mental model for the moving cut is to consider a "wave" propagating in time from the sinks to the sources at a pace of one cut per time step (i.e., the wave is at B_t at time t).

In order to prove that all protocols must use more than T rounds to complete the unicast, we measure the amount of input information (i.e., originating from $\{s_i\}_i$) that can be "ahead" of B_t at time t; i.e., the amount of information available to $V \setminus B_t$. At $t = 0$ this amount is clearly 0, and we are clearly done if we can prove that at $t = T$ that amount does not contain all input information. Formalizing this idea allows us to prove the following lemma.

[1] Such a protocol must send d_i packets along paths of length at most T between each sour-sink pair (s_i, t_i), and it can send at most c_e packets through each edge e during any of the T rounds, or at most $c_e \cdot T$ packets overall.

Lemma 1.6 (Moving cut)**.** *Let $\mathcal{M} = (G, \mathcal{S})$ be a k-unicast instance. Suppose that after increasing each edge e's length from one to $\ell_e \in \mathbb{Z}_{\geq 1}$, we have*

(1) the total length increase, $\sum_{e\in E} c_e(\ell_e - 1)$ is less than $\sum_{i=1}^{k} d_i$, and

(2) any sink is at distance at least T from any source; i.e., $\forall i, j \in [k]$ we have $dist_\ell(s_i, t_j) \geq T$.

Then any *(coding) protocol for $\mathcal{M}$ has makespan at least T.*

Proof. We will show via simulation that a protocol solving $\mathcal{M}$ in at most $T-1$ rounds would be able to compress $\sum_{i=1}^{k} d_i$ random bits to a strictly smaller number of bits, thereby leading to a contradiction. Our simulation proceeds as follows. We have two players, Alice and Bob, who control different subsets of nodes. In particular, if we denote by $A_r \triangleq \{v \in V \mid \min_i \mathrm{dist}_\ell(s_i, v) \leq r\}$ the set of nodes at distance at most r from any source, then during any round $r \in \{0, 1, \ldots, T-1\}$ all nodes in A_r are "spectated" by Alice. By spectated we mean that Alice gets to see all of these nodes' private inputs and received transmissions during the first r rounds. Similarly, Bob, at time r, spectates $B_r \triangleq V \setminus A_r$. Consequently, if at round r a node $u \in V$ spectated by Alice sends a packet to a node $v \in V$, then Bob will see the contents of that packet if and only if Bob spectates the node v at round $r+1$. That is, this happens only if $u \in A_r$ and $v \in B_{r+1} = V \setminus A_{r+1}$. Put otherwise, Bob can receive a packet from Alice along edge e during times $r \in [\min_i \mathrm{dist}_\ell(s_i, u), \min_i \mathrm{dist}_\ell(s_i, v) - 1]$. Therefore, the number of rounds transfer can happen along edge e is at most $\min_i \mathrm{dist}_\ell(s_i, v) - \min_i \mathrm{dist}_\ell(s_i, u) - 1 \leq \ell_e - 1$. Hence, the maximum number of bits transferred from Alice to Bob via e is $c_e(\ell_e - 1)$. Summing up over all edges, we see that the maximum number of bits Bob can ever receive during the simulation is at most $\sum_{e\in E} c_e(\ell_e - 1) < \sum_{i=1}^{k} d_i$. Now, suppose Alice has some $\sum_{i=1}^{k} d_i$ random bits. By simulating this protocol with each source s_i having (a different) d_i of these bits, we find that if all sinks receive their packet in T rounds, then Bob (who spectates all t_j at time $T-1$, as $\min_i \mathrm{dist}_\ell(s_i, t_j) \geq T$ for all j) learns all $\sum_{i=1}^{k} d_i$ random bits while receiving less than $\sum_{i=1}^{k} d_i$ bits from Alice – a contradiction. □

2.2 From Dual Solution to Moving Cut

In the previous section we showed that high objective value for the primal LP implies an upper bound on the routing time for the given instance. In this section we show that low objective value of the primal LP — implying a feasible dual LP solution of low value — yields a moving cut for some sub-instance, allowing us to relate the fastest routing protocol to the fastest coding protocol.

By definition, a low-value feasible solution to the dual LP $\textsc{Cut}_{\mathcal{M}}(T)$ assigns non-negative lengths $\ell : E \to \mathbb{R}_{\geq 0}$ such that (1) the c-weighted sum of ℓ-lengths is small, i.e., $\sum_{e\in E} c_e \ell_e = \tilde{O}(1/T)$, as well as (2) if h_i is the ℓ-length of the T-hop-bounded ℓ-shortest path between s_i and t_i, then $\sum_{i\in[k]} d_i \cdot h_i \geq 1$.

Our coding lower bound of Lemma 1.6 needs a common lower bound on all $\mathrm{dist}(s_i, t_j)$ and not a lower bound on $\sum_i d_i \cdot \mathrm{dist}(s_i, t_i)$. In particular, to present a moving cut for an instance induced by some subset of sessions with indices $I \subseteq [k]$ we need to lower bound $\mathrm{dist}(s_i, t_i)$ for all $i \in I$. Lemma 2.3, proven in Appendix C using a "continuous" bucketing argument, does just this.

Lemma 2.3. *Given sequences $h_1, \ldots, h_k, d_1, \ldots, d_k \in \mathbb{R}_{\geq 0}$ with $\sum_{i=1}^{k} d_i \cdot h_i \geq 1$ there exists a non-empty subset $I \subseteq [k]$ with $\min_{i\in I} h_i \geq \frac{1}{\alpha_{\mathrm{gap}} \cdot \sum_{i\in I} d_i}$ for $\alpha_{\mathrm{gap}} \in \left[1, O\left(\log \frac{\sum_{i=1}^{k} d_i}{\min_{i=1}^{k} d_i}\right)\right]$.*

We next show that the sub-instance defined by source-sink pairs indexed by I as in Lemma 2.3 nearly satisfies the conditions of a moving cut. Specifically, letting $\alpha_{\mathrm{gap}} = O\left(\log \frac{\sum_{i=1}^{k} d_i}{\min_{i=1}^{k} d_i}\right)$ be as in Lemma 2.3, we prove the following.

Lemma 2.4. *Let $\mathcal{M} = (G, \mathcal{S})$ be a k-unicast instance with feasible solution $\{h_i, \ell_e \mid i \in [k], e \in E\}$ to $\textsc{Cut}_{\mathcal{M}}(T)$ with objective value $T \sum_{e\in E} c_e \ell_e = z$. Then there exist indices $I \subseteq [k]$ and integral edge lengths $\tilde{\ell} : E \to \mathbb{Z}_{\geq 1}$ that satisfy $\sum_{e\in E} c_e(\tilde{\ell}_e - 1) \leq z \cdot \sum_{i\in I} d_i$ and $dist_{\tilde{\ell}}(s_i, t_i) \geq T/\alpha_{\mathrm{gap}}$ for all $i \in I$.*

Proof. Let $I \subseteq [k]$ be a subset of indices as guaranteed by Lemma 2.3. Define $\tilde{\ell}_e \triangleq 1 + \lfloor \ell_e \cdot T \cdot \sum_{i \in I} d_i \rfloor$ for all $e \in E$ and note that $\tilde{\ell}_e \in \mathbb{Z}_{\geq 1}$. By definition of $\tilde{\ell}$ and $T \sum_e c_e \ell_e = z$, we get the first condition,

$$\sum_{e \in E} c_e(\tilde{\ell}_e - 1) \leq \sum_e c_e \ell_e \cdot T \cdot \sum_{i \in I} d_i = z \cdot \sum_{i \in I} d_i.$$

We now show that $\text{dist}_{\tilde{\ell}}(s_i, t_i) > T/\alpha_{\text{gap}}$ for $i \in I$. Consider any simple path p between $s_i \rightsquigarrow t_i$. Denote by $\tilde{\ell}(p)$ and $\ell(p)$ the length of p with respect to $\tilde{\ell}$ and ℓ, respectively. It is sufficient to show that $\tilde{\ell}(p) > T/\alpha_{\text{gap}}$ for all such p. If $p \notin \mathcal{P}_i(T)$, i.e., the hop-length of p is at least $|p| \geq T$. Then $\tilde{\ell}(p) \geq |p| \geq T \geq T/\alpha_{\text{gap}}$ since $\tilde{\ell}_e \geq 1 \ \forall e \in E$ and $\alpha_{\text{gap}} \geq 1$. On the other hand, if $p \in \mathcal{P}_i(T)$, then by our choice of I as in Lemma 2.3 and the definition of h_i, we have that $\ell(p) \geq h_i \geq \frac{1}{\alpha_{\text{gap}} \sum_{i \in I} d_i}$, hence

$$\tilde{\ell}(p) \geq \ell(p) \cdot T \cdot \sum_{i \in I} d_i = \frac{1}{\alpha_{\text{gap}} \cdot \sum_{i \in I} d_i} \cdot T \cdot \sum_{i \in I} d_i = T/\alpha_{\text{gap}}. \qquad \square$$

Finally, to obtain a moving cut we need to move from a bound on $\text{dist}_{\tilde{\ell}}(s_i, t_i)$ for all $i \in I$ to a bound on $\text{dist}_{\tilde{\ell}}(s_i, t_j)$ for all $i, j \in I' \subseteq I$. In Lemma 2.5, a formal statement, proof and discussion of which we defer to Section 2.3, yields precisely such a bound, showing that such I' must contains a large subset of indices $I'' \subseteq I'$, $|I''| \geq |I'|/9$ such that for all $i, j \in I''$ we have $\text{dist}_{\tilde{\ell}}(s_i, t_j)$.

Lemmas 2.4 and 2.5 now allow us to construct, given a feasible dual solution of low value, a moving cut for an instance induced by a subset of the sessions. This implies a lower bound on the makespan of the induced instance – and therefore of the entire instance.

Lemma 2.2. *If the optimal value of* $\textsc{ConcurrentFlow}_{\mathcal{M}}(T)$ *is at most* $z^* \leq 1/10$*, then the coding makespan for* $\mathcal{M}$ *is at least* $T/(C \cdot \log(k) \cdot \log(\sum_i d_i/d_i))$ *for some constant* $C > 0$*.*

Proof. By Lemma 2.4, if $\textsc{ConcurrentFlow}_{\mathcal{M}}(T)$ has value less than $z^* \leq \frac{1}{10} < \frac{1}{9}$, then there exist positive integer edge lengths $\tilde{\ell} : E \to \mathbb{Z}_{\geq 1}$ and a subset of indices $I \subseteq [k]$ satisfying $\sum_e c_e \cdot (\tilde{\ell}_e - 1) < \sum_{i \in I} d_i/9$ and $\text{dist}_{\tilde{\ell}}(s_i, t_i) \geq T/\alpha_{\text{gap}}$ for all $i \in I$. By Lemma 2.5 with the graphic metric defined by $\tilde{\ell}$ and each pair (s_i, t_i) repeated d_i times, this implies the existence of a multiset of indices I' in $[k]$ of size $|I'| \geq \sum_i d_i/9$ such that $\text{dist}_{\tilde{\ell}}(s_i, t_j) \geq T/O(\log k \cdot \alpha_{\text{gap}})$ for all $i, j \in I'$. Thus, taking each pair (s_i, t_i) indexed by I' at least once, we find a subset of sessions $I'' \subseteq [k]$ such that $\sum_{i \in I''} d_i \geq \sum_{i \in [k]} d_i/9$ and $\text{dist}_{\tilde{\ell}}(s_i, t_j) \geq T/O(\log k \cdot \alpha_{\text{gap}})$ for all $i, j \in I''$. But then, by Lemma 1.6, we find that the sub-instance induced by theses sessions has makespan at least $T/O(\log k \cdot \alpha_{\text{gap}})$, and therefore so does $\mathcal{M}$. $\square$

Remark. We note that the $\log k$ term in Lemma 2.2's bound is due to the $\log k$ term in the bound of Section 2.3. For many topologies, including genus-bounded and minor-free networks, this $\log k$ term can be replaced by a constant (see Section 2.3), implying smaller makespan gaps for such networks.

2.3 From Pairwise to All-Pairs Distances

This section is dedicated to a discussion and proof of the following Lemma that seems potentially useful beyond the scope of this paper.

Lemma 2.5. *Let* $(X, dist)$ *be a metric space and let* $T > 0$*. Given* k *pairs* $\{(s_i, t_i)\}_{i \in [k]}$ *of points in* X *with at most* n *distinct points in* $\bigcup_i \{s_i, t_i\}$ *and pairwise distances at least* $dist(s_i, t_i) \geq T$*, there exists a subset of indices* $I \subseteq [k]$ *of size* $|I| \geq \frac{k}{9}$ *such that* $dist(s_i, t_j) \geq \frac{T}{O(\log k)}$ *for all* $i, j \in I$*. Moreover, such a set can be computed in polynomial time.*

We note that the above lemma is similar to the main Theorem of Arora et al. [8]. Our result holds for general metrics with a factor of $O(\log k)$ in the distance loss, while their holds for ℓ_2^2 metrics with a factor of $O(\sqrt{\log k})$. The results are incomparable and both are tight. (The tightness of Lemma 2.5 can be shown to be tight for graph metrics, for example in graph metrics of constant-degree expanders.)

To prove Lemma 2.5 we rely on *padded decompositions* [28]. To define these, we introduce some section-specific notation. Let (X, dist) be a metric space. Let the (weak) diameter of a set of points $U \subseteq X$ be denoted by $\mathrm{diam}(U) \triangleq \max_{x,y\in U} \mathrm{dist}(x, y)$. We say a partition $P = \{X_1, X_2, \ldots, X_t\}$ of X is Δ-*bounded* if $\mathrm{diam}(X_i) \leq \Delta$ for all i. Next, for $U \subseteq X$ and a partition P as above, we denote by $U \subseteq P$ the event that there exists a part $X_i \in P$ containing U in its entirety; i.e., $U \subseteq X_i$. Let $B(x, \rho) \triangleq \{y \in X \mid \mathrm{dist}(x, y) \leq \rho\}$ denote the ball of radius $\rho \geq 0$ around $x \in X$.

Definition 2.6. *Let $(X, dist)$ be a metric space. We say that a distribution $\mathcal{P}$ over Δ-bounded partitions of X is (β, Δ)-*padded *if, for some universal constant δ, it holds that for every $x \in X$ and $0 \leq \gamma \leq \delta$,*

$$\Pr_{P\sim\mathcal{P}}[B(x, \gamma\Delta) \not\subseteq P] \leq \beta\gamma.$$

In words, each part of the partition has diameter at most Δ and the probability of any point x in the metric being at distance less than $\gamma\Delta$ from a different part than its own part is at most $\beta\gamma$. Such decompositions were presented, for example, by Gupta et al. [28].

Lemma 2.7 ([28])**.** *Any metric $(X, dist)$ on k points admits a (β, Δ)-padded decomposition, for any $\Delta > 0$ and some $\beta = O(\log k)$. Such a decomposition can be computed in polynomial time.*

We are now ready to prove Lemma 2.5.

Proof of Lemma 2.5. First note that we can focus on the metric space induced by the k distinct points. Let $\mathcal{P}$ be a Δ-bounded β-padded decomposition with $\Delta \triangleq T - 1$ and $\beta = O(\log k)$. We first note that for all $i \in [k]$, s_i and t_i are contained in different parts since the diameter of each part X_i is at most $\Delta = T - 1$ and $\mathrm{dist}(s_i, t_i) \geq T$. Furthermore, letting $\gamma = \frac{1}{2\beta}$, we have that $\Pr[B(s_i, \gamma\Delta) \subset P] \geq \frac{1}{2}$. Let $I' \subseteq [k]$ be the subset of indices i with $B(s_i, \gamma\Delta) \subseteq P$. Then we have $\Pr[i \in I'] \geq \frac{1}{2}$ for all $i \in [k]$.

Flip a fair and independent head/tails coin for each part in P. Let $U \subseteq X$ be the set of points in parts whose coin came out heads, and $V \subseteq X$ be the analogous set for tails. Then for each $i \in I'$ we have that $\Pr[s_i \in U \text{ and } t_i \in V] = \frac{1}{4}$. Let $I \subseteq I'$ be the subset of indices i with $s_i \in U$ and $t_i \in V$, giving $\Pr[i \in I] \geq \Pr[i \in I'] \Pr[i \in I \mid i \in I'] = \frac{1}{2} \cdot \frac{1}{4} = \frac{1}{8}\ \forall i \in I$. We also have that $\mathrm{dist}(s_i, t_j) > \rho = \frac{T-1}{\beta_{\text{gap}}}$ for all $i, j \in I$, since $B(s_i, \rho) \subseteq U$ for all $i \in I \subseteq I'$ and $\{t_j\}_{j\in I} \cap U = \emptyset$. Therefore, this random process yields a subset of indices $I \subset [k]$ such that $\mathrm{dist}(s_i, t_j) > \frac{T-1}{\beta_{\text{gap}}}$, of expected size at least $\mathbb{E}[|I|] \geq \sum_{i\in k} \Pr[i \in I] \geq \frac{k}{8}$. As $k - \mathbb{E}[|I|]$ is a non-negative random variable, Markov's inequality implies that with constant probability $k - |I| \leq \frac{64}{63} \cdot (k - \mathbb{E}[|I|]) \leq \frac{8k}{9}$. The Lemma follows. ☐

Remark: The $O(\log k)$ term in Lemma 2.5's bound is precisely the smallest possible β for which (β, Δ)-padded decompositions of the metric exist. For many graphic metrics, such as those of minor-excluded, bounded-genus, and bounded-doubling-dimension networks, padded decompositions with smaller β exist [2, 28, 48]. This improves the bounds of Lemma 2.5 and thus Lemma 2.2 by $(\log k)/\beta$, implying the same improvement for our makespan coding gaps for such networks.

3 Polylogarithmic Coding Gap Instances

In this section we construct a family of multiple-unicast instances with polylogarithmic makespan coding gap. More precisely, we construct instances where the coding gap is at least $(5/3)^{2^i}$ and the size (both the number of edges and sessions) is bounded by $2^{2^{O(2^i)}}$. Here we give a bird's eye view of the construction and leave the details to subsequent subsections. We clarify that all big-O bounds like $f = O(g)$ mean there exists a universal constant $c > 0$ s.t. $f \leq c \cdot g$ for all admissible values (in particular, there is no assumption on f or g being large enough).

We use the graph product of [13] as our main tool. Given two multiple-unicast instances I_1, I_2 (called the *outer* and *inner instance*, resp.) we create a new instance I_+ where the coding gap is the product of the coding gaps of I_1 and I_2. The product is guided by a *colored bipartite graph* $B = (V_1, V_2, E)$ where each edge is labeled by $(\chi_1, \chi_2) =$ (edge in I_1, session in I_2). Precisely, we create $|V_1|$ copies of I_1, $|V_2|$ copies of I_2 and for each edge $(a, b) \in E(B)$ with label (χ_1, χ_2) we replace the edge χ_1 in the a^{th} copy of I_1 with session χ_2 in the b^{th} copy of I_2.

To prove a lower bound on the coding gap, one needs to upper bound the coding makespan and lower bound the coding makespan. The former is easy: the coding protocols nicely compose. The latter, however, is more involved. Our main tool is Lemma 1.4, which necessitates (i) keeping track of *cut edges* F along each instance I such that all source-sink pairs of I are well-separated after edges in F are deleted, and (ii) keeping the ratio $r \triangleq \frac{k}{|F|}$, number of sessions to cut edges, high. We must ensure that the properties are conserved in the product instance I_+. For (i), i.e., to disallow any short paths from forming as an unexpected consequence of the graph product, we choose B to have *high girth*. Also, we replace edges F in the outer instances with paths rather than connecting them to a session in the inner instance. Issue (ii) is somewhat more algebraically involved but boils down to ensuring that the ratio of sessions to cut edges (i.e., r) in the inner instance is comparable to the size (i.e., number of edges) of the outer instance itself. Note that makes the size of the outer instance I_1 insignificant when compared to the size of the inner instance I_2.

We recursively define a family of instances by parametrizing them with a "level" $i \geq 0$ and a lower bound on the aforementioned ratio r, denoting them by $I(i, r)$. We start for $i = 0$ with the $5/3$ instance of Figure 1 where we can control the ratio the aforementioned ratio r by changing the number of sessions k (at the expense of increasing the size). Subsequently, an instance on level i is defined as a product two of level $i - 1$ instances with appropriately chosen parameters r. One can show that the coding makespan for a level i instance is at most 5^{2^i} and routing makespan is at least 3^{2^i}, hence giving a coding gap of $(5/3)^{2^i}$. Furthermore, we show that the size of $I(i, r)$ is upper bounded by $r^{2^{O(2^i)}}$, giving us the full result.

Finally, we note an important optimization to our construction and specify in more detail how $I(i, r)$ is defined. Specifically, it is defined as the product of $I_1 \triangleq I(i - 1, 3r)$ being the outer instance and $I_2 \triangleq I(i - 1, m_1/f_1)$ being the inner instance, where m_1 and f_1 are the number of total and cut edges of I_1. This necessitates the introduction and tracking of another parameter $u \triangleq m/f$ to guide the construction. We remark that this might be necessary since if one uses a looser construction of $I_2 \triangleq I(i-1, m_1)$ the end result $I(i, r)$ would be of size $r^{2^{O(i \cdot 2^i)}}$ and give a coding gap of $\exp\left(\frac{\log \log k}{\log \log \log k}\right)$, just shy of a polylogarithm.

3.1 Gap Instances and Their Parameters

In this section we formally define the set of instance parameters we will track when combining the instances.

A **gap instance** $I = (G, \mathcal{S}, F)$ is a multiple-unicast instance $\mathcal{M} = (G, \mathcal{S})$ over a connected graph G, along with an associated set of **cut edges** $F \subseteq E(G)$. We only consider gap instances where the set of terminals is disjoint, i.e., $s_i \neq s_j, s_i \neq t_j, t_i \neq t_j$ for all $i \neq j$. Furthermore, edge capacities and demands are one; i.e., $c_e = 1\ \forall e \in E(G)$, and $d_i = 1\ \forall (s_i, t_i, d_i) \in \mathcal{S}$. A gap instance $I = (G, \mathcal{S}, F)$ has **parameters** (a, b, f, k, m, r, u) when:

- $\mathcal{M}$ admits a network coding protocol with makespan at most a.
- Let $\mathrm{dist}_{G \setminus F}(\cdot, \cdot)$ be the hop-distance in G after removing all the cut edges F. Then for all terminals $i \in [k]$ we have that $\mathrm{dist}_{G \setminus F}(s_i, t_i) \geq b$.
- The number of cut edges is $f = |F|$.

- The number of sessions is $k = |\mathcal{S}|$.
- The graph G has at most m edges; i.e., $|E(G)| \leq m$.
- r is a lower bound on the ratio between number of sessions and cut edges; i.e., $k/f \geq r$.
- u is an upper bound on the ratio between number of total edges and cut edges; i.e., $m/f \leq u$.

We note that the parameters of a gap instance immediately imply a lower bound on the optimal routing makespan via Lemma 1.4. Indeed, all packets transmitted in the first $b-1$ rounds must pass through F, and thus at most $f \cdot (b-1)$ packets can be sent between any source and its sink in the first $b-1$ rounds, implying that under any routing protocol, most sessions have completion time at least b.

Observation 3.1. *Let I be a gap instance with parameters (a, b, f, k, m, r, u) and $b \leq r$. Then the routing makespan for $(G, \mathcal{S})$ is at least b. Moreover, for any routing protocol of I, at least $k \cdot (1 - \frac{b-1}{r})$ sessions have completion time at least b.*

As an application of the above observation, we obtain another proof of the lower bound of the routing makespan for the family of instances of Figure 1. More generally, letting the cut edges be the singleton $F = \{(S, T)\}$, we obtain a family of gap instances with the following parameters.

Fact 3.2. *The family of gap instances of Figure 1 have parameters $(3, 5, 1, k, \theta(k^2), k, \theta(k^2))$ for $k \geq 5$.*

The above family of gap instances will serve as our base gap instances in a recursive construction which we describe in the following section.

3.2 Graph Product of Two Gap Instances

In this section we present the graph product that combines two instances to obtain one a with higher coding gap.

Definition 3.3. ***Colored bipartite graphs*** *are families of bipartite graphs $\mathcal{B}(n_1, n_2, m, k, g)$. Graphs $B = (V_1, V_2, E) \in \mathcal{B}(n_1, n_2, m, k, g)$ are bipartite graphs with $|V_1| = n_1$ (resp. $|V_2| = n_2$) nodes on the left (resp., right), each of degree m (resp., k), and these graphs have girth at least g. In addition, edges of B are colored using two color functions,* ***edge color*** *$\chi_1 : E(B) \to [m]$ and* ***session color*** *$\chi_2 : E(B) \to [k]$, which satisfy the following.*

- *$\forall v \in V_1$, the edge colors of incident edges form a complete set $\{\chi_1(e) \mid e \ni v\} = [m]$.*
- *$\forall v \in V_2$, the session colors of incident edges form a complete set $\{\chi_2(e) \mid e \ni v\} = [k]$.*
- *$\forall v \in V_1$, the session colors of incident edges are unique $|\{\chi_2(e) \mid e \ni v\}| = 1$.*
- *$\forall v \in V_2$, the edge colors of incident edges are unique, i.e, $|\{\chi_1(e) \mid e \ni v\}| = 1$.*

The size of the colored bipartite graphs will determine the size of the derived gap instance obtained by performing the product along a colored bipartite graph. The following gives a concrete bound on the size and, in turn, allows us to control the growth of the gap instances obtained this way.

Lemma 3.4 ([13])**.** *$\forall r, m, g \geq 3$, there exists a colored bipartite graph $B \in \mathcal{B}(n_1, n_2, m, k, 2g)$ with $n_1, n_2 \leq (9mk)^{g+3}$.*

Performing the product along a colored bipartite graph. Having defined colored bipartite graphs, we are now ready to define the graph product of I_1 and I_2 along B.

For $i \in \{1,2\}$ let $I_i = (G_i, \mathcal{S}_i, F_i)$ be a gap instance with parameters $(a_i, b_i, f_i, k_i, m_i, r_i, u_i)$ and let $B \in \mathcal{B}(n_1, n_2, 2(m_1 - f_1), k_2, g)$ be a colored bipartite graph with girth $g \triangleq 2b_1b_2$. We call I_1 the **outer instance** and I_2 the **inner instance**. Denote the **product** gap **instance** $I_+ \triangleq T(I_1, I_2, B)$ by the following procedure:

- Replace each non-cut edge $e = \{u, v\} \in E(G_1) \backslash F_1$ with two anti-parallel arcs $\vec{e} = (u, v), \overleftarrow{e} = (v, u)$ and let $\vec{E}(G_1) = \{\vec{e}_1, \vec{e}_2, \ldots, \vec{e}_{2(m_1 - f_1)}\}$ be the set of all such arcs.
- Construct n_1 copies of $(V(G_1), \vec{E}(G_1))$ and n_2 copies of G_2. Label the i^{th} copy as $G_1^{(i)}$ and $G_2^{(i)}$.
- Every cut edge $e \in F_1$ and every $i \in [n_1]$ replace edge e in $G_1^{(i)}$ by a path of length a_2 with the same endpoints. Let $f_e^{(i)}$ be an arbitrary edge on this replacement path.
- For every $(i,j) \in E(B)$ where $i \in [n_1], j \in [n_2]$ with edge color χ_1 and session color χ_2 do the following. Let $\vec{e}_{\chi_1}^{(i)} = (x, y)$ be the χ_1th arc in $G_1^{(i)}$ and let (s,t) be the χ_2th terminal pair in $G_2^{(j)}$. Merge x with s and y with t; delete $\vec{e}_{\chi_1}^{(i)} = (x, y)$ from $G_1^{(i)}$.
- For each session in the outer instance $(s_i, t_i, d_i = 1) \in \mathcal{S}_1$ add a new session $(s_i^{(j)}, t_i^{(j)}, 1)$ in $G_1^{(j)}$ for $j \in [n_1]$ to the product instance.
- The cut edges F_+ in the product instance I_+ consist of the the union of the following: (i) *one* arbitrary (for concreteness, first one) edge from all of the a_2-length paths that replaced cut edges in $G_1^{(i)}$, i.e., $\{f_e^{(i)} \mid e \in F_1, i \in [n_1]\}$, and (ii) all cut edges in copies of G_2, i.e., $\{e^{(i)} \mid e \in F_2, i \in [n_2]\}$.

We now give bounds on how the parameters change after combining two instances. First, we note that by composing network coding protocols for I_1 and I_2 in the natural way yields a network coding protocol whose makespan is at most the product of these protocols' respective makespans.

Lemma 3.5. *(Coding makespan) The product instance I_+ admits a network coding protocol with makespan at most a_1a_2.*

Less obviously, we show that if we choose a large enough girth g for the colored bipartite graph, we have that the b parameter of the obtained product graph is at least the product of the corresponding b parameters of the inner and outer instances.

Lemma 3.6 (Routing makespan)**.** *Let $I_+ = (G_+, \mathcal{S}_+, F_+)$ be the product instance using a colored bipartite graph B of girth $g \triangleq 2b_1b_2$ and let $dist_{G_+ \backslash F_+}(\cdot, \cdot)$ be the hop-distance in G_+ with all the edges of F_+ deleted. We have that $dist_{G_+ \backslash F_+}(s_i, t_i) \geq \min(b_1b_2, \frac{g}{2}) = b_1b_2$ for all $(s_i, t_i, d_i) \in \mathcal{S}_+$.*

Proof. Let p be a path in $G_+ \setminus F_+$ between some terminals $s_i \rightsquigarrow t_i$ that has the smallest hop-length among all $(s_i, t_i, d_i) \in \mathcal{S}_+$. We want to show that $|p| \geq \min(b_1b_2, \frac{g}{2})$.

First, let q be the path in the colored bipartite graph B that corresponds to p. There are some technical issues with defining q since merging vertices in the graph product has the consequence that some $v \in V(G_+)$ belong to multiple nodes $V(B)$. To formally specify q, we use the following equivalent rephrasing of the graph product that will generate an "expanded instance" G'_+. Instead of "merging" two vertices u, v as in G_+, connect then with an edge e of hop-length $h(e) = 0$ and add e to G'_+. Edges from G_+ have hop-length $h(e) = 1$ and are analogously added to G'_+. The path p can be equivalently specified as the path between $s_i \rightsquigarrow t_i$ in $G'_+ \setminus F_+$ that minimizes the distance $\text{dist}_h(s_i, t_i)$. Now, each vertex $V(G'_+)$ belongs to exactly one vertex $V(B)$, hence the path q in B corresponding to p is well-defined. Note that p is a closed path in G_+ and q is a closed path in B.

Suppose that q spans a non-degenerate cycle in B. Then $|p| \geq \frac{|q|}{2} \geq \frac{g}{2}$, where the last inequality $|q| \geq g$ is due to the girth of G. The first inequality $|p| \geq \frac{|q|}{2}$ is due to the fact that when q enters a node $v \in V_2(B)$, a node representing an inner instance, the corresponding path p had to traverse at least one inner instance edge before its exit since the set of terminals is disjoint and a path can enter/exit inner instances only in terminals.

Suppose now that q does not span a cycle in B, therefore the set of edges in q span a tree $\mathcal{T}$ in B and q is simply the (rotation of the unique) Eulerian cycle of that tree. Notation-wise, let $v \in V_1(B)$ be the node in the colored bipartite graph B that contains the critical terminals s_i and t_i and suppose that $\mathcal{T}$ is rooted in v. If the depth of $\mathcal{T}$ is 1 (i.e., consists only of $v \in V_1(B)$ and direct children $w_1, \ldots, w_t \in V_2(B)$), then p must correspond to a $s_i \rightsquigarrow t_i$ walk in v, where each (non-cut) edge traversal is achieved by a non-cut walk in the inner instance w_j between a set of inner terminals. Note that every $s_i \rightsquigarrow t_i$ non-cut walk has hop-length at least b_1 and each non-cut walk in the inner instance has hop-length at least b_2, for a cumulative $b_1 \cdot b_2$.

Finally, suppose that $\mathcal{T}$ has depth more than 1, therefore there exists two $v, w \in V(\mathcal{T})$ and $v, w \in V_1(B)$. Since $\mathcal{T}$ is traversed via an Eulerian cycle, the path p passes through two terminals of $(s_j, t_j, \cdot) \in \mathcal{S}_+$. Let p' be the natural part of p going from $s_j \rightsquigarrow t_j$, e.g., obtained by clipping the path corresponding to the subtree of q in $\mathcal{S}$. Furthermore, let p'' be the part of the p connecting v and w and is disjoint from p''. From the last paragraph we know that $|p''| \geq 1$ since it passes through at least one $u \in V_2(B)$. Also, by minimality of $s_i \rightsquigarrow t_i$ we have that $\text{dist}_h(s_j, t_j) \geq \text{dist}_h(s_i, t_i)$. Now we have a contradiction since $\text{dist}_h(s_i, t_i) = |p| \geq |p'| + |p''| \geq 1 + \text{dist}_h(s_j, t_j)$. $\square$

Combining Lemmas 3.5 and 3.6 together with some simple calculations (deferred to Appendix D), we find that the product instance is a gap instance with the following parameters.

Lemma 3.7. *For $i \in \{1, 2\}$ let $I_i = (G_i, \mathcal{S}_i, F_i)$ be a gap instance with parameters $(a_i, b_i, f_i, k_i, m_i, r_i, u_i)$ with $\frac{m_i}{f_i} \geq 2$ and $a_i \geq 2$; let $B \in \mathcal{B}(n_1, n_2, 2(m_1 - f_1), k_2, 2b_1b_2)$ be a colored bipartite graph. Then $I_+ \triangleq T(G_1, G_2, B)$ is a gap instance with parameters $a_+ \triangleq a_1a_2$, $b_+ \triangleq b_1b_2$, $f_+ \triangleq n_1f_1 + n_2f_2$, $k_+ \triangleq n_1k_1$, $m_+ \triangleq a_2n_1f_1 + n_2m_2$, $r_+ \triangleq r_1\frac{1}{1+2u_1/r_2}$, $u_+ \triangleq u_2\frac{1+a_2/2}{1+r_2/(2u_1)}$. Moreover, $\frac{m_+}{f_+} \geq 2$ and $a_+ \geq 2$.*

3.3 Iterating the Graph Product

Having bounded the parameters obtained by combining two gap instances, we are now ready to define a recursive family of gap instances from which we obtain our polylogarithmic makespan network coding gap.

Definition 3.8. *We recursively define a collection of gap instances $(I(i, r))_{i \geq 0, r \geq 5}$, and denote its parameters by $(a_{i,r}, b_{i,r}, f_{i,r}, k_{i,r}, m_{i,r}, r_{i,r}, u_{i,r})$. For the base case, we let $I(0, r)$ be the gap instance of Fact 3.2 with parameters $(3, 5, 1, r, \theta(r^2), r, \theta(r^2))$. For $i + 1 > 0$ we define $I(i + 1, r) \triangleq T(I_1, I_2)$. Here, $I_1 \triangleq I(i, 3r)$ and $I_2 \triangleq I(i, u_{i,3r})$, with parameters $(a_1, \ldots, u_1)$ and $(a_2, \ldots, u_2)$, respectively.*

In other words, I_1 is defined such that $r_1 = 3r_+$ and I_2 such that $r_2 = u_1$. In Appendix D we study the growth of the parameters of our gap instance families. Two parameters that are easy to bound for this construction are the following.

Observation 3.9. *For any $i \geq 0$ and $r \geq 5$, we have $a_{i,r} = 3^{2^i}$ and $b_{i,r} = 5^{2^i}$.*

A less immediate bound, whose proof is also deferred to Appendix D, is the following bound on the number of edges of the gap instances..

Lemma 3.10. *We have that $\log m_{i,r} \leq 2^{O(2^i)} \log r$ for all $i \geq 0, r \geq 5$.*

3.4 Lower Bounding the Coding Gap

We are now ready to prove this section's main result – a polylog(k) makespan coding gap.

Theorem 1.3. *There exists an absolute constant $c > 0$ and an infinite family of k-unicast instances whose makespan coding gap is at least $\Omega(\log^c k)$.*

Proof. For each $i \geq 0$ and $r \triangleq 5^{2^i}$, consider $I_{i,r}$ as defined above. By Observation 3.9 this gap instance has coding makespan at most $a_{i,r} = 3^{2^i}$. Moreover, also by Observation 3.9, this instance has $b_{i,r} = 5^{2^i}$, and so by Observation 3.1 its routing makespan is at least 5^{2^i}. Hence the makespan coding gap of $I_{i,r}$ is at least $(5/3)^{2^i}$. It remains to bound this gap in terms of $k \triangleq k_{i.r}$.

As the terminals of $I_{i,r}$ are disjoint, we have that k is upper bounded by the number of nodes of $I_{i,r}$, which is in turn upper bounded by $m_{i,r}$, as $I_{i,r}$ is connected and not acyclic. That is, $k \leq m_{i,r}$. But by Lemma 3.10, we have that $\log m_{i,r} \leq 2^{O(2^i)} \cdot \log r = 2^{O(2^i)} \cdot O(2^i) \leq 2^{O(2^i)} \leq 2^{c' \cdot 2^i}$, for some universal constant $c' > 0$. Therefore, stated in terms of k, the makespan coding gap is at least

$$(5/3)^{2^i} = 2^{2^i \log 5/3} = (2^{c' 2^i})^{\frac{\log 5/3}{c'}} \geq (\log m_{i,r})^c \geq \log^c k,$$

where $c \triangleq \frac{\log 5/3}{c'} > 0$ is a universal constant, as claimed. $\square$

4 Coding Gaps for Other Functions of Completion Times

In this section we extend our coding gap results to other time complexity measures besides the makespan. For our upper bounds, we show that our coding gaps for ℓ_∞ minimization of the completion times (makespan) implies similar bounds for a wide variety of functions, including all weighted ℓ_p norms; proving in a sense that ℓ_∞ is the "hardest" norm to bound. The following lemma underlies this connection.

Lemma 4.1. *Let α be an upper bound on the coding gap for completion times' ℓ_∞ norm (makespan). Then, if multiple-unicast instance $\mathcal{M}$ admits a coding protocol with completion times $(T_1, T_2, \ldots, T_k)$, there exists a routing protocol for $\mathcal{M}$ with completion times placewise at most $(4\alpha \cdot T_1, 4\alpha \cdot T_2, \ldots, 4\alpha \cdot T_k)$.*

Proof. Let $\mathcal{M}$ be a multiple-unicast instance. Let $(T_1, T_2, \ldots, T_k)$ be the vector of completion times of some coding protocol. Without loss of generality, assume $T_1 \leq T_2 \leq \cdots \leq T_k$. Next, for any $j \in \mathbb{Z}$, denote by $\mathcal{M}_j$ the sub-instance of $\mathcal{M}$ induced by the unicasts with completion time $T_i \in [2^j, 2^{j+1})$. Then, there exists a network coding protocol for each $\mathcal{M}_j$ with makespan at most 2^{j+1}. Consequently, there exists a routing protocol for $\mathcal{M}_j$ with makespan at most $\alpha \cdot 2^{j+1}$. Scheduling these protocols in parallel, in order of increasing $j = 0, 1, 2, \ldots$, we find that a unicast with completion time $T_i \in [2^j, 2^{j+1})$ in the optimal coding protocol has completion time in the obtained routing protocol which is at most

$$\sum_{j' \leq j} \alpha \cdot 2^{j'+1} \leq 2\alpha \cdot 2^{j+1} \leq 4\alpha \cdot T_i. \qquad \square$$

Note that unlike our routing protocols for makespan minimization of Theorem 1.2, the proof here is non-constructive, as it assumes (approximate) knowledge of the completion times of each unicast in the optimal coding protocol. Nonetheless, this proof guarantees the existence of a protocol, which suffices for our needs. In particular, applying Lemma 4.1 to the coding protocol minimizing a given weighted ℓ_p norm, we immediately obtain the following.

Corollary 4.2. *Let α be an upper bound on the coding gap for completion times' ℓ_∞ norm (makespan). Then the coding gap for any weighted ℓ_p norm of the completion times is at most 4α.*

Plugging in our coding gap upper bound of Theorem 1.2, we therefore obtain a generalization of Theorem 1.2 to any weighted ℓ_p norm, as well as average completion time (which corresponds to ℓ_1).

Theorem 4.3. *The network coding gap for any weighted ℓ_p norms of completion times is at most*

$$O\left(\log(k) \cdot \log\left(\sum_i d_i / \min_i d_i\right)\right).$$

Note that similar bounds hold even more generally. In particular, for any sub-homogeneous function of degree d (i.e., $f(c \cdot \vec{x}) \leq c^d \cdot f(\vec{x})$, Lemma 4.1 implies a coding gap of at most $(4\alpha)^d$, where α is the best upper bound on the coding gap for makespan minimization.

Lower bounds. As with makespan minimization, a polylogarithmic dependence in the problem parameters as in Theorem 4.3, as we prove below. Crucially, we rely on our makespan coding gap's examples displaying the property that under coding nearly all unicast sessions' completion time is at least polylogarithmically larger than under the best coding protocol.

Theorem 4.4. *There exists an absolute constant $c > 0$ and an infinite family of k-unicast instances whose ℓ_p-coding gap is at least $\Omega(\log^c k)$.*

Proof. We follow the proof of Theorem 1.3 and consider $I_{i,r}$, this time setting $r \triangleq (5^{2^i})^2 = 5^{2^{i+1}}$. This does not change the parameters $a_{i,r} = 3^{2^i}$ and $b_{i,r} = 5^{2^i}$, nor the bound $\alpha \triangleq (5/3)^{2^i} \geq \log^c k$ for some absolute constant $c > 0$. This instance has a coding protocol with completion times $(a_{i,r}, \ldots, a_{i,r})$, and so this coding protocol's completion times' ℓ_p value is $a_{i,r}$. On the other hand, by Observation 3.1, at least $k \cdot (1 - \frac{b_{i,r}}{r}) \geq (1 - o(1)) \cdot k$ pairs have routing completion time at least $b_{i,r}$, where $o(1)$ tends to 0 as $i \to \infty$. Consequently, the ℓ_p-value of any routing protocol's completion times is at least $(1 - o(1)) \cdot b_{i,r}$. Since $b_{i,r}/a_{i,r} = \alpha \geq \log^c k$, we obtain the required coding gap. □

5 Conclusions and Open Questions

In this paper we study completion-time coding gaps; i.e., the ratio for a given multiple-unicast instance, of the fastest routing protocol's completion time to the fastest coding protocol's completion time. We provide a strong characterization of these gaps in the worst case, showing they can be polylogarithmic in the problem parameters, but no greater.

Better bounds (for networks of interest). We show the coding gap is upper bounded by

$$O\left(\log(k) \cdot \log\left(\sum_i d_i / \min_i d_i\right)\right).$$

The necessity of the $\log k$ term is essentially argued by our lower bounds of Section 3, which, among other things, exhibit a polylogarithmic lower bound for the structural result of Section 2.2. On the other hand, we do not know of any compelling reason the $\log\left(\sum_i d_i / \min_i d_i\right)$ term is necessary. If removed, it would yield the same coding gap upper bound for completion time and its special case of throughput maximization. Furthermore, we note that the $\log k$ term in our proof stems from trade-offs of some metric decomposition results. Specifically, $\log k$ is precisely the padding parameter of a padded metric decomposition. The long line of work on padded decomposition implies that this term can be replaced by a constant, yielding an $O\left(\log \sum_i d_i / \min_i d_i\right)$ completion-time coding gap for bounded-minor-free graphs [1, 22, 43], bounded doubling-dimension graphs Gupta et al. [28] and genus-bounded graphs (e.g., planar graphs) [48]. Whether lower completion-time coding gaps can be shown for other graphs of interest, either via better padded decompositions or directly, seems a natural direction to explore.

Implications to other fields. As discussed in Section 1 and Section 1.4, the conjectured non-existence of throughput coding gaps for multiple unicast has been used to prove (conditional) lower bounds in many seemingly-unrelated problems. It would be interesting to see whether our upper and lower bounds on the coding gap for various functions of multiple unicasts' completion time would prove useful in proving *unconditional* lower bounds for other models of computation.

An example of a model where our results imply unconditional lower bounds is the popular CONGEST model from distributed computing [60]. Minimizing the makespan in our communication model exactly corresponds to minimizing the time complexity of distributed algorithms in CONGEST. Furthermore, multiple unicast is an exceedingly important problem in CONGEST, since it is the basis of most of the known approaches for impossibility results [19, 21, 61] and is also used as a subroutine in distributed algorithms (e.g., [26]). An immediate consequence of our work is that one can *lift* a super-polylogarithmic lower bound (impossibility) for routing protocols into an *unconditional* super-polylogarithmic lower bound for *all* protocols. For instance, this can be used to strengthen the lower bounds of Haeupler et al. [32] for multiple unicasts in restricted topologies.

Extensions to Multiple Multicast and Implications. A prominent future direction is to attain similar polylogarithmic coding gap results for multiple *multicasts*. This would lead to a better understanding of an important subroutine used in distributed optimization, called *distributed partwise aggregation*, which is essentially equivalent to multiple multicasts. This subroutine is used in tasks such as computing the minimum spanning tree [25, 31], min-cut [25] and shortest paths [33]. In particular, generalizing our makespan coding gaps to multiple multicast would show that a technique called *low-congestion shortcuts* [25] is *instance-optimal* for partwise aggregation [73].

Acknowledgements The authors would like to thank Mohsen Ghaffari for suggesting an improvement to Theorem 1.2 which resulted in a coding gap independent of n, Anupam Gupta for pointing out a simplification of Lemma 2.5 and the Lemma's similarity to [8, Theorem 1], and Paritosh Garg for bringing [13] to our attention.

Appendix

A Completion Time vs. Throughput

In this section we argue why network coding upper bounds for makespan imply coding gaps for throughput maximization. We first introduce the standard definitions of the throughput maximization model [3, 13]. The differences between the throughput and completion-time model (see Section 1.1) are highlighted in blue.

Throughput maximization model. A *multiple-unicast instance* $\mathcal{M} = (G, \mathcal{S})$ is defined over a communication network, represented by an undirected graph $G = (V, E, c)$ with capacity $c_e \in \mathbb{Z}_{\geq 1}$ for each edge e. The $k \triangleq |\mathcal{S}|$ *sessions* of $\mathcal{M}$ are denoted by $\mathcal{S} = \{(s_i, t_i, d_i)\}_{i=1}^{k}$. The (maximum) *throughput* of $\mathcal{M}$ is the supremum $r > 0$ such that there exists a sufficiently large $b > 0$ where the following problem has a correct protocol. Each session consists of source node s_i, which wants to transmit a packet to its sink t_i, consisting of $\lceil r \cdot b \cdot d_i \rceil$ sub-packets (e.g., an element of an underlying field). A *protocol* for a multiple-unicast instance is conducted over finitely-many *synchronous time steps*. Initially, each source s_i knows its packet, consisting of d_i sub-packets. At any time step, the protocol instructs each node v to send a different packet along each of its edges e. The packet contents are computed with some predetermined function of packets received in prior rounds by v or originating at v. The total number of sub-packets sent through an edge e over the duration of the entire protocol is at most $b \cdot c_e$. We differentiate the maximum throughput achievable by coding and routing protocols as r^C and r^R, respectively. The throughput coding gap is the largest ratio r^C/r^R that can be achieved for any instance.

Relating completion times and throughput. The throughput maximization intuitively corresponds to the makespan minimization of an instance with asymptotically-large packet sizes. More formally, we modify a multiple unicast instance $\mathcal{M}$ by increasing its demands by a factor of w while keeping the capacities the same. This causes the makespan to increase. We argue that the slope of the increase with respect to w is exactly the throughput of $\mathcal{M}$.

Definition A.1. *Given a multiple-unicast instance $\mathcal{M}$ we define $C^C(w)$ and $C^R(w)$ to be the makespan of the fastest coding and routing protocols when the all demands are multiplied by a common factor w.*

Observation A.2. *Let $\mathcal{M}$ be a multiple-unicast instance. The maximum throughput r corresponding to $\mathcal{M}$ is equal to $\sup_{w\to\infty} w/C(w)$ for both coding and routing. Formally, $r^C = \sup_{w\to\infty} w/C^C(w)$ and $r^R = \sup_{w\to\infty} w/C^R(w)$.*

Proof. We drop the R/C superscripts since the proof holds for both without modification. Let $L \triangleq \sup_{w\to\infty} w/C(w)$. We first argue that $L \geq r$, i.e., we can convert a throughput protocol to a makespan-bounded one. For simplicity, we will assume that $b = 1$ (b from the throughput definition); when this is not the case one needs to appropriately re-scale the sub-packets for the completion-time protocol.

Let $\mathcal{T}$ be a protocol of throughput at least $r - o(1)$ and let T be the total number of rounds $\mathcal{T}$ uses (note that in the throughput setting T has no impact on the quality of $\mathcal{T}$). Let $w \in \mathbb{Z}$ be a sufficiently large number. We use pipelining by scheduling $w' \triangleq w/(r - o(1))$ independent copies of $\mathcal{T}$: the first one starting at time 1, second at time 2, ..., last one at time w'. Each copy operates on a separate set of sub-packets, with the pipelined protocol being able to transmit $(r - o(1)) \cdot d_i \cdot w' = d_i \cdot w$ sub-packets across the network (in line with Definition A.1) in at most $w' + T$ rounds. Note that $\mathcal{T}$ sends at most c_e sub-packets over an edge e over its entire execution, hence the pipelined version of $\mathcal{T}$ never sends more than c_e sub-packets during any one round. In other words, we have that $C(w) \leq w' + T$. Letting $w \to \infty$ (which implies $w' \to \infty$), we have that

$$L \geq \frac{w}{C(w)} \geq \frac{w}{w' + T} = (r - o(1))\frac{w'}{w' + T} = r - o(1).$$

We now argue the converse $r \geq L$, i.e., we can convert a completion-time protocol into a throughput protocol with the appropriate rate. The result essentially follows by definition. By assumption, for some sufficiently large $w > 0$ there exists a protocol with makespan at most $C(w) \leq w/(L - o(1))$. The protocol sends a total of at most $C(w)c_e$ sub-packets over an edge e. Furthermore, by construction of $C(w)$, each source-sink pair successfully transmits $w \cdot d_i$ sub-packets. By noting that $w \cdot d_i = (L - o(1))C(w)d_i$, we conclude that by using $b \triangleq C(w)$ we get a protocol with rate $L - o(1)$. □

Corollary A.3. *Suppose that the makespan coding gap is at most α (over all instances). Then the throughput coding gap is at most α.*

Proof. Consider some multiple unicast instance $\mathcal{M}$, with coding throughput r. By Observation A.2, for sufficiently large w there is a coding protocol $\mathcal{P}_1$ satisfying $w/C^C(w) \geq r - o(1)$, i.e., $C^C(w) \leq w/(r - o(1))$. By the makespan coding gap assumption, there exists a routing protocol $\mathcal{P}_2$ implying that $C^R(w) \leq \alpha \cdot w/(r - o(1))$. Furthermore, following the proof of Observation A.2, protocol $\mathcal{P}_2$ implies a routing throughput of r' for the original instance $\mathcal{M}$, satisfying

$$r' \geq w/C^R(w) \geq (r - o(1))/\alpha = r/\alpha - o(1).$$

In other words, $r/r' \geq \alpha + o(1)$ and we are done. □

B Network Coding Model for Completion Time

In this section we formalize the k-session unicast communication problem and the notion of completion time for it. We note that our model is not new—e.g., it is equivalent to the models of Chekuri et al. [16], Wang and Chen [68] that study delay in communication networks.

The input to a *k-session unicast problem* $(G, \mathcal{S})$ consists of a graph $G = (V, E, c)$, where edges have capacities $c : E \to \mathbb{R}_{\geq 0}$, and a set of k sessions $\mathcal{S} = \{(s_i, t_i, d_i)\}_{i=1}^k$. Each triplet (s_i, t_i, d_i) corresponds to the source $s_i \in V$, sink $t_i \in V$ and the demand $d_i \in \mathbb{R}_{\geq 0}$ of session i. The graph G can be either directed or undirected, where in the latter case we model an undirected edge e as two directed edges $\vec{e}, \overleftarrow{e}$ where both of them have equal capacity $c(\vec{e}) = c(\overleftarrow{e}) = c_e$.[2]

Each source s_i is privy to an input message $m_i \in M_i$ generated by an arbitrary stochastic source with entropy at least d_i, hence the entropy of the random variable m_i is d_i. The sources corresponding to different sessions are independent.

A T-round *network coding computation* consists of a set of $|E| \times T$ coding functions $\{f_{\vec{e},r} : M \to \Gamma\}_{\vec{e} \in E, 1 \leq r \leq T}$, where Γ is some arbitrary alphabet and $M \triangleq \prod_i M_i$. These functions satisfy the following properties:

- The entropy of any coding function $f_{\vec{e},r}$ never exceeds the edge capacity $c(\vec{e})$, i.e., $H(f_{\vec{e},r}) \leq c(\vec{e})$ for all $\vec{e} \in E, 1 \leq r \leq T$.
- For each directed edge $\vec{e} = (u, v) \in E$ and round $1 \leq r \leq T$ the function $f_{\vec{e},r}$ is computable from communication history received strictly before round r at node u. In other words, let the communication history $Y_{u,r}$ be defined as $\{m_i \mid i \in [k], s_i = u\} \cup \{f_{(x,y),r'} \mid y = u \text{ and } r' < r\}$, then $H(f_{(u,v),r} | Y_{u,r}) = 0$.
- The completion times of a network coding computation are $(T_1, T_2, \ldots, T_k) \in \mathbb{Z}_{\geq 0}^k$ when the following holds. For every session i, the message m_i of the session (s_i, t_i, d_i) must be computable from the sink t_i's history after T_i rounds are executed, i.e., $H(m_i | Y_{t_i, T_i+1}) = 0$.

Remark: The above "bare-bones" formalization is sufficient for all of our results to hold. However, such a formalization can be unwieldy since a complete instance description would also need to specify a stochastic distribution corresponding to each source s_i. A standard way of avoiding this issue is to simply assume the sources generate a uniformly random binary string of length d_i (forcing d_i to be an integer). Without going into too much detail, we mention this assumption can be made without loss of generality if we allow for (1) an arbitrarily small decoding error $\varepsilon > 0$, (2) slightly perturbing the edge capacities c_e and source entropies d_i by ε, and (3) scaling-up both c_e's and d_i's by a common constant $b > 0$; this approach is standard in the literature (e.g., see [3, 13]).

C Deferred Proofs of Section 2

In this section we provide proofs deferred from Section 2, starting with the proof of Lemma 2.1, restated here for ease of reference.

Lemma 2.1. *Let $z \leq 1$, $\{f_i(p) \mid i \in [k], p \in \mathcal{P}_i(T)\}$ be a feasible solution for* $\textsc{ConcurrentFlow}_{\mathcal{M}}(T)$. *Then there exists an integral routing protocol with makespan $O(T/z)$.*

To prove the above, we rely on the celebrated $O(\text{congestion} + \text{dilation})$ packet scheduling theorem of Leighton et al. [49]. In particular, we use the solution to $\textsc{ConcurrentFlow}_{\mathcal{M}}(T)$ to obtain a

[2] Papers such as Adler et al. [3] often impose an alternative condition $c(\vec{e}) + c(\overleftarrow{e}) = c(e)$, which would make our proofs slightly heavier on notation. However, their convention can only impact the results up to a factor of 2, which we typically ignore in this paper.

collection of short paths with bounded congestion (i.e., bounded maximum number of paths any given edge belongs to). We then route along these paths in time proportional to these paths' maximum length and congestion. The issue is that the feasible LP solution provides fractional paths, hence requiring us to round the LP solution. Independent rounding would result in paths of length T and congestion $T/z + O(\log n)$ (with high probability). To avoid this additive dependence on n, we rely on the following theorem of Srinivasan and Teo [67].

Lemma C.1 ([67], Theorem 2.4, paraphrased)**.** *Let $\mathcal{M}$ be a multiple-unicast instance and for each $i \in [k]$ let $\mathcal{D}_i$ be a distribution over $s_i \rightsquigarrow t_i$ paths of hop-length at most L. Suppose that the product distribution $\prod \mathcal{D}_i$ has expected congestion for each edge at most L. Then there exists a sample $\omega \in \prod \mathcal{D}_i$ (i.e., a choice of a from $\mathcal{D}_i$ between each $s_i \rightsquigarrow t_i$) with (maximum) congestion $O(L)$.*

Using the above lemma to round the LP and using Leighton et al. [49] path routing to route along the obtained paths yields Lemma 2.1.

Proof of Lemma 2.1. Consider an optimal solution to this $\textsc{ConcurrentFlow}_{\mathcal{M}}(T)$. Clearly, picking for each pair (s_i, t_i) some d_i paths in $P_i(T)$ with each $p \in \mathcal{P}_i(T)$ picked with probability $f_i(p) \cdot d_i / \sum_{p \in \mathcal{P}_i(T)} f_i(p) \leq f_i(p)/z$ yields an expected congestion at most $T \cdot c_e/z$ for each edge e. That is, thinking of G as a multigraph with c_e copies per edge, each such parallel edge has congestion T/z. On the other hand, each such path has length at most $T \leq T/z$ (since $z \leq 1$). Therefore, by Lemma C.1, there exist choices of paths for each pair of (maximum) congestion and hop-bound (i.e., dilation) at most $O(T/z)$. But then, using $O(\text{congestion} + \text{dilation})$ routing [49] this implies an integral routing protocol with makespan $O(T/z)$, as claimed. □

Here we prove Lemma 2.3, restated here for ease of reference.

Lemma 2.3. *Given sequences $h_1, \ldots, h_k, d_1, \ldots, d_k \in \mathbb{R}_{\geq 0}$ with $\sum_{i=1}^k d_i \cdot h_i \geq 1$ there exists a non-empty subset $I \subseteq [k]$ with $\min_{i \in I} h_i \geq \frac{1}{\alpha_{\text{gap}} \cdot \sum_{i \in I} d_i}$ for $\alpha_{\text{gap}} \in \left[1, O\left(\log \frac{\sum_{i=1}^k d_i}{\min_{i=1}^k d_i}\right)\right]$.*

Proof. Suppose (without loss of generality) that $h_1 \geq h_2 \geq \ldots h_k$ and assume for the sake of contradiction that none of the sets $[1], [2], \ldots, [k]$ satisfy the condition. In other words, if we let $d([j]) \triangleq \sum_{i=1}^j d_j$, then $h_i < \frac{1}{\alpha} \cdot \frac{1}{d([i])}$ for all $i \in [k]$. Multiplying both sides by d_i and summing them up, we get that $1 \leq \sum_{i=1}^k d_i h_i < \frac{1}{\alpha} \sum_{i=1}^k \frac{d_i}{d([i])}$. Reordering terms, this implies $\sum_{i=1}^k \frac{d_i}{d([i])} > \alpha$.

Define $f(x)$ as $1/d_1$ on $[0, d_1)$; $1/(d_1 + d_2)$ on $[d_1, d_1 + d_2)$; ...; $1/d([i])$ on $[d([i-1]), d([i]))$ for $i \in [k]$. Now we have

$$\int_0^{d([k])} f(x) = \frac{d_1}{d_1} + \frac{d_2}{d_1 + d_2} + \frac{d_3}{d_1 + d_2 + d_3} + \ldots + \frac{d_k}{d([k])}.$$

However, since $f(x) \leq 1/x$

$$\begin{aligned} \int_0^{d([k])} f(x) &= \int_0^{d_1} f(x)\, dx + \int_{d_1}^{d([k])} f(x)\, dx \\ &\leq 1 + \int_{d_1}^{d([k])} \frac{1}{x}\, dx = 1 + \ln \frac{d([k])}{d_1}. \end{aligned}$$

Hence by setting $\alpha \triangleq 1 + \ln \frac{d([k])}{d_1}$ we reach a contradiction and finish the proof. □

D Deferred Proofs of Section 3

Here we provide the deferred proofs of lemmas of Section 3, restated below for ease of reference.

Lemma 3.5. *(Coding makespan) The product instance I_+ admits a network coding protocol with makespan at most $a_1 a_2$.*

Proof. Suppose there exists a network coding protocol with makespan $t_i \le a_i$ that solves $(G_i, \mathcal{S}_i)$ for $i \in \{1, 2\}$. Functionally, each round in the outer instance $(G_1, \mathcal{S}_1)$ consists of transmitting c_e bits of data from u to v for all arcs (u, v) where $\{u, v\} \in E(G_1)$. This is achieved by running the full t_2 rounds of the inner instance protocol over all copies of the instances which pushes d_i bits from s_i to t_i for all (s_i, t_i, d_i) and all copies of the inner instance. The reason why such inner protocol pushes the information across each arc (u, v) is because u is merged with some s_i, v is merged with t_i, and with $d_i = c_e$ for some $(s_i, t_i, d_i) \in \mathcal{S}_2$ and some copy of the inner instance. In conclusion, by running t_1 outer rounds, each consisting of t_2 inner rounds, we get a $t_1 t_2 \le a_1 a_2$ round protocol for the product instance. □

Lemma 3.7. *For $i \in \{1, 2\}$ let $I_i = (G_i, \mathcal{S}_i, F_i)$ be a gap instance with parameters $(a_i, b_i, f_i, k_i, m_i, r_i, u_i)$ with $\frac{m_i}{f_i} \ge 2$ and $a_i \ge 2$; let $B \in \mathcal{B}(n_1, n_2, 2(m_1 - f_1), k_2, 2b_1 b_2)$ be a colored bipartite graph. Then $I_+ \triangleq T(G_1, G_2, B)$ is a gap instance with parameters $a_+ \triangleq a_1 a_2$, $b_+ \triangleq b_1 b_2$, $f_+ \triangleq n_1 f_1 + n_2 f_2$, $k_+ \triangleq n_1 k_1$, $m_+ \triangleq a_2 n_1 f_1 + n_2 m_2$, $r_+ \triangleq r_1 \frac{1}{1+2u_1/r_2}$, $u_+ \triangleq u_2 \frac{1+a_2/2}{1+r_2/(2u_1)}$. Moreover, $\frac{m_+}{f_+} \ge 2$ and $a_+ \ge 2$.*

Proof of Lemma 3.7. First, the set of terminals in the product instance I_+ is disjoint, as distinct terminals of copies of the outer instance I_1 have their edges associated with distinct terminals source-sink pairs of the inner instance I_2. Consequently, no two terminals of the outer instance are associated with the same node of the same copy of an inner instance. The capacities and demands of I_+ are one by definition. We now turn to bounding the gap instance's parameters.

Parameters a_+ and b_+ are directly argued by Lemma 3.5 and Lemma 3.6. Furthermore, f_+, k_+, m_+ are obtained by direct counting, as follows.

Recall that the cut edges of the outer instance get replaced with a path of length a_2. Since there are n_1 copies of outer instances, each having f_1 cut edges, this contributes $a_2 n_1 f_1$ edges to m_+. The non-cut edges of the outer instance get deleted and serve as a merging directive, hence they do not contribute to m_+. Finally, each edge of the inner instance gets copied into I_+, contributing $n_2 m_2$ as there are n_2 copies of the inner instance.

For r_+ we need to show it is a lower bound on k_+/f_+. We note that $|E(B)| = n_1 \cdot 2(m_1 - f_1) = n_2 k_2$ and proceed by direct calculation:

$$\begin{aligned}\frac{k_+}{f_+} &= \frac{n_1 k_1}{n_1 f_1 + n_2 f_2} = \frac{k_1}{f_1} \cdot \frac{1}{1 + \frac{n_2}{n_1}\frac{f_2}{f_1}} \ge \frac{k_1}{f_1} \cdot \frac{1}{1 + \frac{2(m_1 - f_1)}{k_2}\frac{f_2}{f_1}} \\ &\ge \frac{k_1}{f_1} \cdot \frac{1}{1 + \frac{2m_1}{f_1}\frac{f_2}{k_2}} \ge \frac{k_1}{f_1} \cdot \frac{1}{1 + \frac{2u_1}{r_2}} = r_1 \cdot \frac{1}{1 + \frac{2u_1}{r_2}} = r_+.\end{aligned}$$

For u_+ we need to show it is an upper bound on m_+/f_+. Note that $k_2 \le m_2$ since the set of terminals is disjoint and the graph is connected.

$$\begin{aligned}\frac{m_+}{f_+} &= \frac{n_2 m_2 + a_2 n_1 f_1}{n_2 f_2 + n_1 f_1} = \frac{m_2}{f_2} \cdot \frac{1 + a_2 \frac{n_1}{n_2}\frac{f_1}{m_2}}{1 + \frac{n_1}{n_2}\frac{f_1}{f_2}} \le u_2 \cdot \frac{1 + a_2 \frac{k_2}{2(m_1 - f_1)}\frac{f_1}{m_2}}{1 + \frac{k_2}{2(m_1 - f_1)}\frac{f_1}{f_2}} \\ &\le u_2 \cdot \frac{1 + a_2 \frac{k_2}{2(m_1/f_1 - 1)}\frac{1}{m_2}}{1 + \frac{f_1}{2m_1}\frac{k_2}{f_2}} \le u_2 \cdot \frac{1 + a_2 \cdot \frac{1}{2} \cdot 1}{1 + \frac{1}{2}\frac{r_2}{u_1}} = u_+.\end{aligned}$$

Here the last inequality relies on $m_1/f_1 \geq 2$ and on $k_2 \leq m_2$, which follows from the set of terminals being disjoint and the graph G_2 being connected.

For the final technical conditions, note that $a_+ \geq 2$ is clear from $a_+ = a_1 a_2 \geq 4 \geq 2$. Finally, $\frac{m_+}{f_+} \geq 2$ follows from the following.

$$\frac{m_+}{f_+} = \frac{a_2 n_1 f_1 + n_2 m_2}{n_1 f_1 + n_2 f_2} = a_2 \frac{n_1 f_1}{n_1 f_1 + n_2 f_2} + \frac{m_2}{f_2} \frac{n_2 f_2}{n_1 f_1 + n_2 f_2} \geq 2 \left(\frac{n_1 f_1}{n_1 f_1 + n_2 f_2} + \frac{n_2 f_2}{n_1 f_1 + n_2 f_2} \right) = 2. \square$$

D.1 Upper Bounding $m_{i,r}$

For readability, we sometimes write $u(i,r)$ instead of $u_{i,r}$ and similarly for $m(i,r)$. Also, we note that the technical conditions $a_{i,r} \geq 2$ and $\frac{m_{i,r}}{f_{i,r}} \geq 2$, which clearly hold for $i = 0$, hold for all $i > 0$, due to Lemma 3.7. Finally, we note that by Lemma 3.7, since $r_2 = u_1$ and $a_2 \geq 1$, we have that $u_+ \geq u_2$ and so for all gap instances in the family we have $u_{i,r} \geq u_{i-1,u(i-1,3r)} \geq 5$.

Lemma D.1. *The parameter of $I(i,r)$ for any $i \geq 0, r \geq 5$ satisfy the following.*

(i) $\frac{k_{i,r}}{f_{i,r}} \geq r$,

(ii) $u_{i+1,r} \leq 3^{2^i} \cdot u_{i,u(i,3r)}$ *and*

(iii) $\log m(i+1,r) \leq O(5^{2^{i+1}}) \cdot \log(m_{i,3r} \cdot m_{i,u(i,3r)})$.

Proof. Claim (i) follows from Lemma 3.7, as follows.

$$\frac{k_{i+1,r}}{f_{i+1,r}} \geq r_{i+1,r} = r_{i,3r} \cdot \frac{1}{1 + 2u_{i,3r}/r_{i,u(i,3r)}} \geq 3r \cdot \frac{1}{1 + 2u_{i,3r}/u_{i,3r}} = 3r \cdot \frac{1}{3} = r.$$

We now prove claims (ii) and (iii). Fix i, r and define $I_1 \triangleq I(i, 3r)$ (with parameters $(a_1, \ldots, u_1)$) and $I_2 \triangleq I(i, u_{i,r})$ (with parameters $(a_2, \ldots, u_2)$). We have $u(i+1,r) = u_2 \frac{1+a_2/2}{1+r_2/(2u_1)} \leq u_2 \frac{1+a_2/2}{1+1/2} \leq u_2 \cdot a_2$ (Lemma 3.7), with $a_2 = a_{i,u(i,3r)} = 3^{2^i}$ and $u_2 = u(i, u(i,3r))$ from the iterated tensoring process. Therefore, we conclude that $u(i+1,r) \leq 3^{2^i} \cdot u_{i,u(i,3r)}$, as claimed.

We now prove Claim (iii). The corresponding colored bipartite graph $B \in \mathcal{B}(n_1, n_2, 2(m_1 - f_1), k_2, 2b_1 b_2)$ used to produce the product $I_{i,r}$ has $\max(n_1, n_2) \leq (2(m_1 - f_1)k_2)^{O(b_1 b_2)}$, by Lemma 3.4. Therefore, as $k_2 \leq m_2$, we have that $\max(n_1, n_2) \leq (m_1 \cdot m_2)^{O(b_1 b_2)}$. This implies the following recurrence for $m_{i,r}$.

$$m_{i+1,r} = a_2 n_1 f_1 + n_2 m_2 \leq a_2 \max(n_1, n_2) m_1 m_2.$$

Taking out logs, we obtain the desired bound.

$$\begin{aligned}
\log m_{i+1,r} &\leq \log a_2 + \log \max(n_1, n_2) + \log m_1 m_2 \\
&= O(2^i) + O(b_1 b_2) \log(m_1 m_2) + \log(m_1 m_2) \\
&= O(2^i) + O(5^{2^i}) \log(m_1 m_2) \\
&= O(5^{2^i}) \cdot \log(m_{i,3r} \cdot m_{i,u(i,3r)}).
\end{aligned}$$

$\square$

Given Lemma D.1 we obtain the bound on $u_{i,r}$ in terms of i and r.

Lemma D.2. *We have that $\log u_{i,r} \leq 2^{O(2^i)} \log r$ for all $i \geq 0, r \geq 5$.*

Proof. By Lemma D.1, we have the recursion $u(i+1,r) \leq 3^{2^i} \cdot u(i, u(i, 3r))$ with initial condition $u(0,r) = O(r^2)$, by Fact 3.2. Taking out logs, we obtain $\log u(i+1,r) \leq O(2^i) + \log u(i, u(i,3r))$ and $\log u(0,r) = O(\log r)$. We prove via induction that $\log u(i,r) \leq \frac{1}{c}(c^2)^{2^i} \cdot \log r$ for some sufficiently large $c > 0$. In the base case $\log u(0,r) = O(\log r) \leq \frac{1}{c}(c^2) \log r = c \log r$. For the inductive step we have:

$$\begin{aligned}
\log u(i+1,r) &\leq O(2^i) + \log u(i, u(i,3r)) \\
&\leq O(2^i) + \frac{1}{c}(c^2)^{2^i} \log u(i,3r) \\
&\leq O(2^i) + \frac{1}{c}(c^2)^{2^i} \frac{1}{c} c^{2^i} \log 3r \\
&\leq O(2^i) + \frac{1}{c^2}(c^2)^{2^{i+1}} \log 3r \\
&\leq \frac{1}{c}(c^2)^{2^{i+1}} \log r,
\end{aligned}$$

where the last inequality holds for $i \geq 1$ and $r \geq 5$ and a sufficiently large $c > 0$. ☐

Plugging in the bound of Lemma D.2 and Lemma D.1 we can prove inductively the upper bound on the number of edges of $I_{i,r}$ in terms of i and r given by Lemma 3.10, restated here.

Lemma 3.10. *We have that $\log m_{i,r} \leq 2^{O(2^i)} \log r$ for all $i \geq 0, r \geq 5$.*

Proof. By Lemma D.1, we have the recursion $\log m(i+1,r) \leq O(5^{2^{i+1}}) \cdot \log(m_{i,3r} \cdot m_{i,u(i,3r)})$ with initial condition $\log m(0,r) = O(\log r)$, by Fact 3.2. We prove via induction that $\log m_{i,r} \leq c^{2^i} \log r$ for a sufficiently large universal constant $c > 0$. In the base case $\log m_{0,r} \leq O(\log r) \leq c \log r = c^{2^0} \log r$. For the inductive step, using Lemma D.2 to bound $\log u(i,3r)$, we have:

$$\begin{aligned}
\log m_{i+1,r} &\leq O(5^{2^{i+1}}) \cdot (\log m_{i,3r} + \log m_{i,u(i,3r)}) \\
&\leq O(5^{2^{i+1}}) \cdot \left(c^{2^i} \log 3r + c^{2^i} \log u(i,3r)\right) \\
&\leq O(5^{2^{i+1}}) \cdot \left(c^{2^i} \log 3r + c^{2^i} 2^{O(2^i)} \log 3r\right) \\
&= c^{2^i} \cdot O(5^{2^{i+1}}) \cdot 2^{O(2^i)} \log 3r \\
&\leq c^{2^i+1} \log r,
\end{aligned}$$

where the last inequality holds for $i \geq 1$ and $r \geq 5$ and a sufficiently large $c > 0$. ☐

References

[1] Abraham, I., Gavoille, C., Gupta, A., Neiman, O., and Talwar, K. 2014. Cops, robbers, and threatening skeletons: Padded decomposition for minor-free graphs. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing (STOC)*. 79–88.

[2] Abraham, I., Gavoille, C., Gupta, A., Neiman, O., and Talwar, K. 2019. Cops, robbers, and threatening skeletons: Padded decomposition for minor-free graphs. *SIAM Journal on Computing (SICOMP) 48,* 3, 1120–1145.

[3] Adler, M., Harvey, N. J., Jain, K., Kleinberg, R., and Lehman, A. R. 2006. On the capacity of information networks. In *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*. Society for Industrial and Applied Mathematics, 241–250.

[4] Afshani, P., Freksen, C. B., Kamma, L., and Larsen, K. G. 2019. Lower bounds for multiplication via network coding. 10:1–10:12.

[5] Agarwal, A. and Charikar, M. 2004. On the advantage of network coding for improving network throughput. In *Information Theory Workshop, 2004. IEEE*. 247–249.

[6] Ahlswede, R., Cai, N., Li, S.-Y., and Yeung, R. W. 2000. Network information flow. *IEEE Transactions on Information Theory 46,* 4, 1204–1216.

[7] Al-Bashabsheh, A. and Yongaçoglu, A. 2008. On the k-pairs problem. In *Proceedings of the IEEE International Symposium on Information Theory (ISIT)*. 1828–1832.

[8] Arora, S., Rao, S., and Vazirani, U. 2009. Expander flows, geometric embeddings and graph partitioning. *Journal of the ACM (JACM) 56,* 2, 5.

[9] auf der Heide, F. M. and Vöcking, B. 1999. Shortest-path routing in arbitrary networks. *Journal of Algorithms 31,* 1, 105–131.

[10] Aumann, Y. and Rabani, Y. 1998. An o (log k) approximate min-cut max-flow theorem and approximation algorithm. *SIAM Journal on Computing (SICOMP) 27,* 1, 291–301.

[11] Bertsimas, D. and Gamarnik, D. 1999. Asymptotically optimal algorithms for job shop scheduling and packet routing. *Journal of Algorithms 33,* 2, 296–318.

[12] Blasiak, A., Kleinberg, R., and Lubetzky, E. 2011. Lexicographic products and the power of non-linear network coding. In *Proceedings of the 52nd Symposium on Foundations of Computer Science (FOCS)*. 609–618.

[13] Braverman, M., Garg, S., and Schvartzman, A. 2017. Coding in undirected graphs is either very helpful or not helpful at all. In *Proceedings of the 8th Innovations in Theoretical Computer Science Conference (ITCS)*. 18:1–18:18.

[14] Busch, C., Magdon-Ismail, M., Mavronicolas, M., and Spirakis, P. 2004. Direct routing: Algorithms and complexity. In *Proceedings of the 12th Annual European Symposium on Algorithms (ESA)*. 134–145.

[15] Chekuri, C., Gupta, A., Newman, I., Rabinovich, Y., and Sinclair, A. 2006. Embedding k-outerplanar graphs into l1. *SIAM Journal on Discrete Mathematics 20,* 1, 119–136.

[16] Chekuri, C., Kamath, S., Kannan, S., and Viswanath, P. 2015. Delay-constrained unicast and the triangle-cast problem. In *2015 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 804–808.

[17] Chekuri, C., Khanna, S., and Shepherd, F. B. 2005. Multicommodity flow, well-linked terminals, and routing problems. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing (STOC)*. 183–192.

[18] Chekuri, C., Shepherd, F. B., and Weibel, C. 2013. Flow-cut gaps for integer and fractional multiflows. *Journal of Combinatorial Theory, Series B 103,* 2, 248–273.

[19] Das Sarma, A., Holzer, S., Kor, L., Korman, A., Nanongkai, D., Pandurangan, G., Peleg, D., and Wattenhofer, R. 2012. Distributed verification and hardness of distributed approximation. *SIAM Journal on Computing (SICOMP) 41,* 5, 1235–1265.

[20] Deb, S., Médard, M., and Choute, C. 2006. Algebraic gossip: A network coding approach to optimal multiple rumor mongering. *IEEE/ACM Transactions on Networking (TON) 14,* SI, 2486–2507.

[21] Elkin, M. 2006. An unconditional lower bound on the time-approximation trade-off for the distributed minimum spanning tree problem. *SIAM Journal on Computing (SICOMP) 36,* 2, 433–456.

[22] Fakcharoenphol, J. and Talwar, K. 2003. An improved decomposition theorem for graphs excluding a fixed minor. In *Proceedings of the 6th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX)*. 36–46.

[23] Farhadi, A., Hajiaghayi, M., Larsen, K. G., and Shi, E. 2019. Lower bounds for external memory integer sorting via network coding. In *Proceedings of the 51st Annual ACM Symposium on Theory of Computing (STOC)*. To Appear.

[24] Fragouli, C., Widmer, J., and Le Boudec, J.-Y. 2008. Efficient broadcasting using network coding. *IEEE/ACM Transactions on Networking (TON) 16,* 2, 450–463.

[25] Ghaffari, M. and Haeupler, B. 2016. Distributed algorithms for planar networks II: Low-congestion shortcuts, mst, and min-cut. In *Proceedings of the 27th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 202–219.

[26] Ghaffari, M. and Li, J. 2018. New distributed algorithms in almost mixing time via transformations from parallel algorithms. In *Proceedings of the 32nd*. 31:1–31:16.

[27] Goel, A. and Khanna, S. 2008. On the network coding advantage for wireless multicast in euclidean space. In *Proceedings of the 7th international conference on Information processing in sensor networks*. IEEE Computer Society, 64–69.

[28] Gupta, A., Krauthgamer, R., and Lee, J. R. 2003. Bounded geometries, fractals, and low-distortion embeddings. In *Proceedings of the 44th Symposium on Foundations of Computer Science (FOCS)*. 534.

[29] Haeupler, B. 2015. Simple, fast and deterministic gossip and rumor spreading. *Journal of the ACM (JACM) 62,* 6, 47.

[30] Haeupler, B. 2016. Analyzing network coding (gossip) made easy. *Journal of the ACM (JACM) 63,* 3, 26.

[31] Haeupler, B., Izumi, T., and Zuzic, G. 2016a. Low-congestion shortcuts without embedding. In *Proceedings of the 35th ACM Symposium on Principles of Distributed Computing (PODC)*. 451–460.

[32] Haeupler, B., Izumi, T., and Zuzic, G. 2016b. Near-optimal low-congestion shortcuts on bounded parameter graphs. In *Proceedings of the 30th*. 158–172.

[33] Haeupler, B. and Li, J. 2018. Faster distributed shortest path approximations via shortcuts. *arXiv preprint arXiv:1802.03671*.

[34] Halperin, E., Kortsarz, G., Krauthgamer, R., Srinivasan, A., and Wang, N. 2007. Integrality ratio for group steiner trees and directed steiner trees. *SIAM Journal on Computing (SICOMP) 36,* 5, 1494–1511.

[35] Harvey, D. and Van Der Hoeven, J. 2019. Polynomial multiplication over finite fields in time $O(n \log n)$.

[36] Harvey, N. J., Kleinberg, R., and Lehman, A. R. 2006. On the capacity of information networks. *IEEE/ACM Transactions on Networking (TON) 14,* SI, 2345–2364.

[37] Harvey, N. J., Kleinberg, R. D., and Lehman, A. R. 2004. Comparing network coding with multicommodity flow for the k-pairs communication problem. Tech. rep.

[38] Ho, T., Médard, M., Koetter, R., Karger, D. R., Effros, M., Shi, J., and Leong, B. 2006. A random linear network coding approach to multicast. *IEEE Transactions on Information Theory 52,* 10, 4413–4430.

[39] Hu, T. C. 1963. Multi-commodity network flows. *Operations research 11,* 3, 344–360.

[40] Huang, J., Yin, X., Zhang, X., Du, X., and Li, Z. 2013. On space information flow: Single multicast. In *2013 International Symposium on Network Coding (NetCod)*. IEEE, 1–6.

[41] Jain, K., Vazirani, V. V., Yeung, R., and Yuval, G. 2006. On the capacity of multiple unicast sessions in undirected graphs. *IEEE/ACM Transactions on Networking (TON) 14,* SI, 2805–2809.

[42] Katti, S., Rahul, H., Hu, W., Katabi, D., Médard, M., and Crowcroft, J. 2006. Xors in the air: Practical wireless network coding. In *ACM SIGCOMM computer communication review*. Vol. 36. ACM, 243–254.

[43] Klein, P., Plotkin, S. A., and Rao, S. 1993. Excluded minors, network decomposition, and multicommodity flow. In *Proceedings of the 25th Annual ACM Symposium on Theory of Computing (STOC)*. 682–690.

[44] Koch, R., Peis, B., Skutella, M., and Wiese, A. 2009. Real-time message routing and scheduling. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*. Springer, 217–230.

[45] Kramer, G. and Savari, S. A. 2006. Edge-cut bounds on network coding rates. *Journal of Network and Systems Management 14,* 1, 49.

[46] Krauthgamer, R., Lee, J. R., and Rika, H. 2019. Flow-cut gaps and face covers in planar graphs. In *Proceedings of the 30th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 525–534.

[47] Langberg, M. and Médard, M. 2009. On the multiple unicast network coding, conjecture. In *47th Annual Allerton Conference on Communication, Control, and Computing*. 222–227.

[48] Lee, J. R. and Sidiropoulos, A. 2010. Genus and the geometry of the cut graph:[extended abstract]. In *Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 193–201.

[49] Leighton, F. T., Maggs, B. M., and Rao, S. B. 1994. Packet routing and job-shop scheduling in O(congestion+ dilation) steps. *Combinatorica 14,* 2, 167–186.

[50] Leighton, T., Maggs, B., and Richa, A. W. 1999. Fast algorithms for finding o (congestion+ dilation) packet routing schedules. *Combinatorica 19,* 3, 375–401.

[51] Li, Z. and Li, B. 2004a. Network coding in undirected networks. In *Conference on Information Systems and Sciences (CISS)*.

[52] Li, Z. and Li, B. 2004b. Network coding: The case of multiple unicast sessions. In *Allerton Conference on Communications*. Vol. 16. 8.

[53] Li, Z., Li, B., and Lau, L. C. 2009. A constant bound on throughput improvement of multicast network coding in undirected networks. *IEEE Transactions on Information Theory 55,* 3, 1016–1026.

[54] Linial, N., London, E., and Rabinovich, Y. 1995. The geometry of graphs and some of its algorithmic applications. *Combinatorica 15,* 2, 215–245.

[55] Lovett, S. 2014. Linear codes cannot approximate the network capacity within any constant factor. In *Electronic Colloquium on Computational Complexity (ECCC)*. Vol. 21. 141.

[56] Okamura, H. and Seymour, P. D. 1981. Multicommodity flows in planar graphs. *Journal of Combinatorial Theory, Series B 31,* 1, 75–81.

[57] Ostrovsky, R. and Rabani, Y. 1997. Universal $O(\mathit{congestion} + \mathit{dilation} + \log^{1+\epsilon} n)$ local control packet switching algorithms. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing (STOC)*. Vol. 29. 644–653.

[58] Peis, B., Skutella, M., and Wiese, A. 2009. Packet routing: Complexity and algorithms. In *Proceedings of the 7th Workshop on Approximation and Online Algorithms (WAOA)*. 217–228.

[59] Peis, B. and Wiese, A. 2011. Universal packet routing with arbitrary bandwidths and transit times. In *Proceedings of the 13th Conference on Integer Programming and Combinatorial Optimization (IPCO)*. 362–375.

[60] Peleg, D. 2000. *Distributed computing.* Vol. 5.

[61] Peleg, D. and Rubinovich, V. 2000. A near-tight lower bound on the time complexity of distributed minimum-weight spanning tree construction. *SIAM Journal on Computing (SICOMP) 30,* 5, 1427–1442.

[62] Plotkin, S. and Tardos, É. 1995. Improved bounds on the max-flow min-cut ratio for multicommodity flows. *Combinatorica 15,* 3, 425–434.

[63] Rabani, Y. and Tardos, É. 1996. Distributed packet switching in arbitrary networks. In *Proceedings of the 28th Annual ACM Symposium on Theory of Computing (STOC)*. Vol. 96. 366–375.

[64] Rao, S. 1999. Small distortion and volume preserving embeddings for planar and euclidean metrics. In *Proceedings of the 15th Symposium on Computational geometry (SoCG)*. 300–306.

[65] Rothvoss, T. 2013. A simpler proof for $O(\mathit{Congestion}+\mathit{Dilation})$ packet routing. In *Proceedings of the 16th Conference on Integer Programming and Combinatorial Optimization (IPCO)*. 336–348.

[66] Scheideler, C. 2006. *Universal routing strategies for interconnection networks.* Vol. 1390. Springer.

[67] Srinivasan, A. and Teo, C.-P. 2001. A constant-factor approximation algorithm for packet routing and balancing local vs. global criteria. *SIAM Journal on Computing (SICOMP) 30,* 6, 2051–2068.

[68] Wang, C.-C. and Chen, M. 2016. Sending perishable information: Coding improves delay-constrained throughput even for single unicast. *IEEE Transactions on Information Theory 63,* 1, 252–279.

[69] Wu, Y., Chou, P. A., and Kung, S.-Y. 2005. Minimum-energy multicast in mobile ad hoc networks using network coding. *IEEE Transactions on communications 53,* 11, 1906–1918.

[70] Xiahou, T., Li, Z., Wu, C., and Huang, J. 2014. A geometric perspective to multiple-unicast network coding. *IEEE Transactions on Information Theory 60,* 5, 2884–2895.

[71] Yin, X., Li, Z., Liu, Y., and Wang, X. 2018. A reduction approach to the multiple-unicast conjecture in network coding. *IEEE Transactions on Information Theory 64,* 6, 4530–4539.

[72] Zosin, L. and Khuller, S. 2002. On directed steiner trees. In *Proceedings of the 13th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 59–63.

[73] Zuzic, G. 2019. Distributed optimization beyond worst-case topologies.