Summer 8-2017

Approximation Algorithms for Faster Communication and Cheaper Networks Using Linear Programming

Jennifer Iglesias
Carnegie Mellon University

Carnegie Mellon University

MELLON COLLEGE OF SCIENCE

THESIS

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF Doctor of Philosophy
in Algorithms, Combinatorics and Optimization

TITLE Approximation Algorithms for Faster Communication and Cheaper Networks Using Linear Programming

PRESENTED BY Jennifer Iglesias

ACCEPTED BY THE DEPARTMENT OF Mathematical Sciences

Ramamoorthi Ravi	August 2017
MAJOR PROFESSOR	DATE
Thomas Bohman	August 2017
DEPARTMENT HEAD	DATE

APPROVED BY THE COLLEGE COUNCIL

Rebecca Doerge	August 2017
DEAN	DATE

Approximation Algorithms for faster communication and cheaper networks using Linear Programming

Jennifer Iglesias

August 18, 2017

Department of Mathematical Sciences
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
R. Ravi, Advisor
Anupam Gupta
Po-Shen Loh
Rajmohan Rajaraman

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Keywords: approximation algorithms, communication problems, network design, linear programming, poise, radio model, telephone model, gossip, publisher-subscriber model, tree augmentation problem

In memory of my grandfather, James Iglesias. He taught me to always use my brain.

Abstract

As we are currently in the information age, people expect access to information to exist by default. In order to facilitate the communication of knowledge, efficient networks must be built. In particular, the networks must be built satisfying some constraints while minimizing cost or time. These constraints often make these problems NP-hard. In this thesis, we investigate two different sets of problems: communicating information quickly and building cheap networks. In the communication problems, the goal is to minimize the number of rounds of communication. In the network design problems the goal is to construct a network of minimum cost.

First, we study the communication problem of computing a minimum time schedule to spread rumors in a given graph under the telephone, radio, and wireless (also called edge-star) models. In all of these communication models, the communication happens in discrete rounds. In the telephone model, a matching is given every round and matched nodes exchange all their information with each other. In the radio model, any subset of nodes can broadcast in a round but only nodes with a single broadcasting neighbor get a message. If a node has two or more neighbors broadcasting, they receive no messages due to interference. In the wireless model, any subset of nodes can broadcast in a round and a non-broadcasting node can tune to listen to exactly one of its neighbors. The various rumor spreading problems assume a message at several or all nodes and that each message must reach some target node or set of nodes. The goal is to deliver all messages to their destinations in a minimum number of rounds. The various problems we study include gossip, multicommodity, and asymmetric multicommodity. In the gossip problem, the goal is to communicate messages from every node to every other node. In multicommodity, pairs of nodes are given must exchange their messages with each other. In asymmetric multicommodity, ordered pairs of nodes are given where the source node must send their message to the sink node. Our work exposes relationships across these models and problems and opens up several new avenues for future study. We present a comprehensive study of

the approximation algroithms for these problems. We show various reductions from harder to easier models for special demands. In addition, we develop the first sub-linear approximation algorithm for the asymmetric multicommodity problem in the edge-star model. Using a reduction from minimium induced matchings, we show the minimum radio gossip time has a hardness of approximation of $\Omega(n^{\frac{1}{2}-\epsilon})$ in an n-node graph. We continue studying these communication problems in the case where the host graph is planar. To aid us, we first study the poise problem, where the goal is to build a subgraph which minimizes the sum of the diameter of the subgraph and the maximum degree of the subgraph. We show poise has only a $O(\frac{\log n}{\log \log n})$ integrality gap. This is the first upper bound on the integrality gap of the natural LP; all previous algorithms for poise had yielded only approximations with respect to the integer optimum. Using the poise result and shortest-path separators in planar graphs, we give a poly-logrithmic approximation algorithm for the telephone multicommodity problem. We also use shortest-path separators to show a poly-logarithmic approximation for the radio gossip problem. This is the first result on radio gossip which does not rely on the maximum degree of the graph. Lastly, we show that these results on planar graphs naturally extend to minor-free families of graphs.

Next we turn to the problem of building minimum-cost networks. From the publish-subscribe systems of the early days of the Internet to the recent emergence of Web 3.0 and the Internet of Things, new problems arise in the design of networks centered at producers and consumers of constantly evolving information. In a typical problem, each terminal builds a physical network in the form of a tree or a star centered at it to push or pull its information from. Some publisher-subscriber pairs of nodes need to coordinate which requires that their corresponding networks overlap at some node. These pairs form the demands which must be satisfied. This simple overlap constraint, along with the requirement that the network is a tree or a star, leads to a variety of new questions on the design of overlapping networks. In this thesis, we show that with metric costs and a complete demand graph, approximation algorithms with small constant performance ratios exist regardless of whether the networks built must be trees or stars. For the general demand case, we show that a natural LP formulation has a non-constant integrality gap which indicates potential hardness of approximation. Finally, we show that the general demand version has a logarithmic-factor approximation algorithm by adapting previously known methods.

Lastly, we study the tree augmentation problem (TAP), where the goal is to augment an existing connected network so that the network becomes 2-edge connected with a minimum cost set of edges. A graph is 2-edge connected if the removal of any one edge does not disconnect the graph. First, we show that in weighted TAP, we can restrict our attention to trees which are binary and all the non-tree edges go between two leaves of the tree. We give two different top-down coloring algorithms. Both of our algorithms differ from known techniques for obtaining a $\frac{3}{2}$-approximation in unweighted TAP and current attempts to reach a $\frac{3}{2}$-approximation in weighted TAP. The first algorithm we describe always gives a 2-approximation starting from any feasible

fractional solution to the natural tree cut covering LP. When a fractional solution's non-zero edges are all at least α, then this algorithm achieves a $\frac{2}{1+\alpha}$-approximation. We propose a new conjecture on extreme points of LP relaxations for the problem, which if true, will lead to a potentially constructive proof of an integrality gap of at most $\frac{3}{2}$ for weighted TAP. In the second algorithm, we introduce simple extra valid constraints to the tree edge covering LP. In this algorithm, we focus on deficient edges, which are edges that get covered to an extent less than $\frac{4}{3}$ in the fractional solution. We show that in the support of extreme points for this LP, deficient edges occurs in node-disjoint paths in the tree. When the number of such paths is at most two, we give a top-down coloring algorithm which decomposes $\frac{3}{2}$ times the fractional solution into a convex combination of integer solutions. We believe our algorithms will be useful in eventually resolving the integrality gap of linear programming formulations for TAP. We also investigate a variant of TAP where each edge in the solution must be covered by a cycle of length three (triangle). We give a $\Theta(\log n)$-approximation algorithm for this problem in the weighted case and a 4-approximation in the unweighted case.

Acknowledgments

Chapters 4, 2, and 3 were all based on joint work with Rajmohan Rajaraman, R. Ravi, and Ravi Sundaram [46, 47, 48]. Chapter 5 is based on joint work with R. Ravi [49]. I would also like to thank Robert Carr and László Végh for their numerous discussions related to Chapter 5

Contents

List of Figures

List of Tables

Chapter 1

Introduction

As the world grows and technology gets better, more and more information must be exchanged. This creates problems in both how to build networks to support the connections cheaply and given networks how to communicate information across them quickly. In building networks or network design problems, the goal is to build a network that supports all the constraints while having minimum cost. In communication problems, the goal is to set up a schedule for how information should be sent across a network so that all information is received as quickly as possible.

Both of these types of problems give rise to many NP-hard problems. NP-hard problems are believed to have no polynomial time exact algorithms. In other words, there is no fast way to solve the problem exactly. As many of these problems need to be solved they are relaxed in some way. Many times it is possible to find a feasible solution which is not too much more expensive or longer than the optimal solution. Approximation algorithms are algorithms which find a feasible solution with a guarantee on how far away its cost is from the cost of the optimal solution [82]. For approximation algorithms, a performance guarantee is given; this guarantee is how close to optimal the solution provided is. In particular, a k-approximation algorithm provides a solution which is at most k times the cost of the optimal solution. In this thesis, we provide many new approximation algorithms for both communication and network design problems. In addition, we also give lower bounds; these lower bounds prove that no approximation algorithm (or no approximation algorithms of a given type) can guarantee a solution better than a certain ratio.

1.1 Faster Communication

One set of problems we study in this thesis are problems where the goal is to deliver messages between terminals in the fewest number of rounds. We will look at three different models of communication; radio, edge-star, and telephone. The model of communication determines how messages can be delivered in a single round of communication, but all of our models communicate messages in rounds and hence the goal is to minimize the total number of rounds within which all the messages have been delivered to their required destinations. All these models of communication use a communication network, which is a connected graph where an edge represents two nodes which can communicate directly in one round. In the telephone model, a matching is chosen in the

communication network every round and the paired nodes exchange all their known messages with each other. In the radio model, a subset of nodes broadcast every round and a non-broadcasting node will hear the message from its broadcasting neighbor if it has exactly one broadcasting neighbor. The radio model helps capture the concept of interference, as when more than one neighbor broadcasts the node is unable to acquire any new messages. The edge-star model is similar to the radio model in that some subset of nodes broadcasts at every round, but a non-broadcasting node can choose which neighbor it listens to. The edge-star model is more closely related to how wireless networks function. In addition, we will look at different sorts of demands for how the messages must be delivered; broadcast, gossip, multicommodity, and asymmetric multicommodity. In broadcast, one node must deliver its message to everyone. In gossip, everyone wants to deliver their message to everyone. For multicommodity, pairs of nodes are given and every pair of nodes must exchange their messages. In asymmetric multicommodity, some ordered pairs are given where the first node must deliver its message to the second node.

In Chapter 2 we start the investigation of rumor spreading in a network. In the rumor spreading problem, a schedule is built determining which nodes broadcast and which nodes listen in each round. The goal in this problem is to transmit messages between some pairs of nodes in a minimal number of rounds. In this chapter, we provide approximation algorithms for multiple communication problems in the edge-star model. In the gossip case, the goal is to deliver a message from every node to every other node. In the multicommodity case, we consider where we are given some pairs (s_i, t_i) such that s_i's message must be delivered to t_i. We then further break the multicommodity case further into two cases; asymmetric and symmetric. In the symmetric case, the pairs of nodes want to exchange information, in other words, if (s_i, t_i) is a pair then (t_i, s_i) is also a pair. In the asymetric case, there is no restriction on the structure of the demands. In Chapter 2, we derive the first approximation algorithms for edge-star in both of these cases. We also investigated the radio model, this model more closely models radio communication and interference occurs when multiple neighboring terminals try to broadcast their messages at the same time. We derive a lower bound of approximation for the gossip problem in the radio model.

In Chapter 3, we continue to investigate rumor spreading in structured graphs such as planar graphs. We first study the telephone model, in this case disjoint pairs of terminals can exchange all their information in a given round. In our study of this problem, we also investigate the problem of poise. The poise of a graph is the maximum of its maximum degree and its diameter. In the poise-problem, the goal is to build a network of smallest poise. Given a low poise graph, then we can easily propagate messages through the graph losing only a small factor. In this chapter, we also explore the radio model on the gossip case in planar graphs. In this case, we achieve a poly-logarithmic approximation which is the first approximation for this problem which doesn't depend on the maximum degree.

1.2 Cheaper Networks

In this thesis, we also study the construction of cheap networks. In these problems, the goal is to build a network or networks of the cheapest cost which satisfy some connectivity requirements. In the first problem, we will have every terminal build its own network. Depending on the demands,

different pairs of terminals will want their networks to overlap. In the second problem, you start with a connected network (such as a tree) and want to add the cheapest set of edges so the resulting graph is 2-edge connected. This means that between any two nodes there are two paths.

In Chapter 4, we investigate the publisher-subscriber problem. In the publisher-subscriber problem, the problem will provide some demands (pairs of terminals), and a host graph. The goal is to build a network for every terminal of a specified type, either a tree or a star rooted at the terminal. A demand is satisfied if the networks built for its terminals overlap (share a node). This is to model push-pull models where some of the nodes are publishers, and push out all their information to all nodes in their network, and then the subscriber nodes pull the information in from all the nodes in their network. In this problem, we consider the case where the demands have certain structures. In particular, in the case with complete demands the optimal solution has a nice structure and can easily be found in polynomial time. In the case where the demands form a complete bipartite graph, then we give a constant approximation. For general demands, we give an $O(\log n)$-approximation and show that the natural LP has a gap of at least $\Omega(\log \log n)$ where n is the number of vertices in the graph.

In Chapter 5, we study the tree augmentation probelem (TAP). In TAP, the goal is to transform a connected network into one which is two-edge connected by adding a minimal cost set of links. This problem is based on the problem of adding redundancy to a network so that single failure will not cause the network to become disconnected. There have been many efforts to create a better than 2-approximation algorithm for weighted TAP. We give two different $3/2$-approximation algorithms for special cases of weighted TAP. With one of these approaches, we propose a conjecture which if true would give a $3/2$-approximation algorithm for weighted TAP. In addition, we examine a new linear program for TAP and give a third approach with this LP.

Also, in Chapter 5, we study the 3-cycle tree augmentation problem (3TAP) variant. In this variant, every edge must be in a cycle of size three, as opposed to just in a cycle. In this variant we give an $O(\log n)$-approximation algorithm and show a matching lower bound for the weighted case where n is the number of nodes. In the unweighted case, we show that every minimal solution is a 4-approximation.

1.3 Techniques and Approaches

Throughout this thesis, we rely on linear programming to provide us with an optimal fractional solution. We start by producing an integer program which captures our problem and then relax the integrality constraint to obtain a linear program. We can solve the linear program to find an optimal fractional solution. Any optimal fractional solution will be cheaper than an optimal integer solution, so we will often use the optimal fractional solution as a lower bound on the cost of the optimal integer solution. We use a variety of techniques to convert this fractional solution into in an integer solution. One simple technique we frequently use is randomized rounding. In this case, we will randomly choose parts of the solution to maintain using probabilities based on the fraction that part was used. We use LP rounding to approximate poise in Section 3.2. We also use a deterministic LP rounding approach to achieve a $O(\log n)$ approximation for the general case of the publisher-subscriber problem in Section 4.2.2.

Another common technique we use is reductions to previously known problems. Reductions were first used by Karp when he proved the existence of an NP-hard problem [53]. These reductions allow us to achieve better approximations, better lower bounds, and better integrality gaps. Many of these reductions are also approximation preserving, this means that any improvements to the approximations on the problem reduced to also provides an improved approximation to these problems. We reduce to previously well-studied problems to achieve multiple cases of complete bipartite and complete demands in the publisher-subscriber problem in Sections 4.2.2 and 4.3. We also use reductions in the study of the edge-star and radio communication models; we use a reduction for hardness for radio gossip in Section 2.2 and a reduction for approximation algorithms in multiple cases of edge-star in Sections 2.3.1, 2.3.2, and 2.3.3. In addition, we use a creative reduction from group steiner tree to show that the natural LP for the publisher-subscriber problem has a $\Omega(\log \log n)$ integrality gap 4.2.1.

Some of the algorithms described in this thesis are novel and built directly for the problem they solve. One such problem is the complete demand case for Tree-tree DON in Section 4.5. In this case, we use a flow based proof to show that there exists an optimal solution of a specific form. From there, we are able to check the linear number of cases that are remaining to find the optimal solution. Another algorithm built directly for the problem is our $\tilde{O}(n^{2/3})$-approximation for edge-star asymmetric multicast in Section 2.3.4. In this algorithm, we create two algorithms: a local algorithm and a greedy algorithm. We run the local algorithm to start and then once the number of remaining demands is small enough then we switch to a greedy algorithm.

Another major technique we use is decomposing a fractional solution into a combination of integer solutions. This technique provides not only an approximation algorithm but also gives an upper bound on the integrality of the LP that the fractional solutions satisfy [9]. This technique is used in two special cases of weighted TAP in Sections 5.3 and 5.4. These techniques are of particular interest because they differ from all known better than 2-approximations for unweighted TAP and from all attempts at a better than 2-approximation for weighted TAP. This algorithm in 5.3 leads us to a unique conjecture on the structure of the LP. If this conjecture were true, this would be the first generalization of Jain [50] of its form.

For Chapter 3, we study planar graphs and minor-free families. We use the special structure that they have separators that are a small number of shortest paths which gives a nice way to decompose the problem we are studying among recursive pieces. In particular, planar graphs have path-separators consisting of three shortest paths and minor-free families have path separators consisting of a constant number of paths [1]. In Section 3.3, we solve an LP and find a path separator. We combine the results of these two parts to create our approximation algorithm. In Section 3.4, the approximation algorithm only relies on the existence of path separators with few paths to run. In addition, we extend both of the previous results from planar graphs to minor-free families in Section 3.5.

1.4 Roadmap

In Chapters 2 and 3, we study the communication problems. In particular, Chapter 2 provides an in-depth study of the edge-star model and also provides the lower bound on radio gossip. Chapter 3

focuses on the communication problems on planar graphs and also provides the first bound on the integrality gap for poise. In Chapters 4 and 5, we study the network design problems. Chapter 4 provides approximation algorithms for the different variants of the publisher-subscriber problem along with a new integrality gap lower bound for the problem. Chapter 5 provides multiple new approaches to be used in achieving a better than 2-approximation for TAP. Lastly, Chapter 6 goes over the major results and presents open problems.

Chapter 2

Rumors

2.1 Introduction

Problems modeling rumor spread are central to the design of coordination networks that seek to keep demand pairs of vertices in contact over time. The prototypical example is the *broadcast* problem where a message in a root node must be sent to all the other nodes via connections represented by an undirected graph. We assume that communication proceeds in synchronized rounds. When more than one message is being disseminated, we assume that in each round each node can transmit an unlimited number of messages in one communication. A subset generalization of broadcast is called the *Multicast* problem: a subset of nodes is specified as terminals and the goal is to spread the rumor from the root only to this subset, using other non-terminal nodes if needed in the process. An all-to-all generalization of the broadcast problem is termed *gossip*: every node has its own piece of information that must be communicated to all nodes, and the goal is to have all the information spread to all the nodes in the minimum number of rounds. Gossip and broadcast are special cases of a more general demand model that we may call *multicommodity multicast*: in this most general version, we are given a set of source-sink pairs so that each source has a rumor that must be sent to the corresponding sink. Recall that messages from many sources can all be aggregated and exchanged in one round between any pair that can communicate, and the goal is to minimize the number of rounds. In this chapter, we will study a specialization of the multicommodity demand model called the *symmetric multicommodity* where for every source-sink pair, we also have the symmetric requirment that the sink wants to send its rumor to the source; thus, the demand pairs are unordered in this case. The more general version will be called the *asymmetric multicommodity* demand model to distinguish it from the symmetric demands case.

2.1.1 Models: Telephone, Radio, and Edge-Star, a New Model from Wireless

Different communication models result in different constraints on the set of edges on which messages can be transmitted in a single round. The two most widely studied models are the telephone and radio models: In the **telephone** model, in each round, a node can communicate with at most

one other node, thus the edges on which communication occurs is a matching; In the **radio** model, a set of transmitters broadcast the message out but only their neighbors who are adjacent to exactly only one transmitter can successfully receive the message (while interference prevents other neighbors from receiving the message): the set of edges through which the messages are sent in any round in this model is a set of stars centered at the transmitters, where each leaf of each star has that star's center as its unique neighbor among all the star centers.

In this chapter, we expand the study of rumor spreading problems by introducing a new model based on wireless communications between nodes, which we call the **edge-star** model. We assume that during each round of wireless communication, each transmitter can choose its own channel or frequency distinct from that of all other transmitters. The input undirected graph represents pairs of nodes that are within wireless range of each other. Receiving nodes that are in the vicinity of many different transmitting nodes can choose to tune into the frequency of one of them. In this way, the set of edges in which communication happens in every round is a set of stars which are defined by a subset of edges of the input graph. Note that unlike the radio model, there is no requirement that a receiver be adjacent to exactly one transmitter. This model more closely models wireless networks, where a machine may be able to see many wireless networks, but only interacts with one of these networks at a time.

2.1.2 Previous Work

The radio broadcast and gossip problems have been extensively studied (see the work reviewed in the survey [36]). The best-known scheme for radio broadcast is by Kowalski and Pelc [59] which completes in time $O(D + \log^2 n)$, where n is the number of nodes, and D is the diameter of the graph and is a lower bound to get the message across the graph from any root. The $O(\log^2 n)$ term is also unavoidable as demonstrated by Alon et al. [3] in an example with constant diameter that takes $\Omega(\log^2 n)$ rounds for an optimal broadcast scheme to complete. Elkin and Korsartz [20] also show that this additive log-squared term is best possible unless $NP \subseteq DTIME(n^{\log\log n})$.

The best bound for radio gossip known so far, however, is $O(D + \Delta \log n)$ steps in an n-node graph with diameter D and maximum degree Δ [37]. The maximum degree is not a lower bound on the gossip time, and indeed no previous results are known about the approximability for radio gossip, which is mentioned as an open problem in [36].

In the telephone model, the first poly-logarithmic approximation for minimum broadcast time was achieved by Ravi [70] and the current best known approximation ratio is $O(\frac{\log n}{\log\log n})$ due to Elkin and Korsartz [24]. The best known lower bound on the approximation ratio for telephone broadcast is $3 - \epsilon$ [22].

In his study of the telephone broadcast time problem, Ravi [70] introduced the idea of finding low poise spanning trees to accomplish broadcast: the *poise* of a spanning tree of an undirected graph is the sum of its diameter and its maximum degree. In the course of deriving a poly-logarithmic approximation, Ravi also showed how a tree of poise P in an n-node graph can be used to complete broadcast starting from any node in $O(P \cdot \frac{\log n}{\log\log n})$ steps - we will use this observation later.

Nikzad and Ravi [66] studied the telephone multicommodity multicast problem, and gave the first sub-linear approximation algorithm with performance ratio $2^{O(\log\log k\sqrt{\log k})}$ for instances with

k source-sink pairs.

Gandhi et al. [33] recently studied the Radio Aggregation Scheduling problem which is a gathering version of the rumor spreading problem in the radio model. The set of edges in which communication occurs in every round is a matching with the additional property that if the edges within receivers and within senders are ignored, the communicating edges form an induced matching. In this model they prove a tight $\Theta(n^{1-\epsilon})$-approximation for their radio aggregation scheduling. Our results were derived independently of their methods.

2.1.3 Our contributions

We give the first results on the approximability of gossip and multicommodity multicast problems in the radio model. We introduce the edge-star model based on wireless channels and give the first approximation results for minimum time rumor spreading by relating them to their analogs in the telephone model.

1. We show that it is NP-hard to approximate gossip in the radio model within a factor of $O(n^{1/2-\epsilon})$ in an n-node graph. This result is derived by isolating a gathering version of the broadcast problem in the radio model and relating it in a simple bipartite graph to induced matchings (Section 2.2).
2. We obtain an $O(\frac{\log n}{\log \log n})$ approximation algorithm for gossip in the edge-star model by reducing the problem to the broadcast problem in the telephone model (Section 2.3.1).
3. We consider the special case where the underlying graph is a tree, and show that the multicommodity multicast in the edge-star model reduces to the broadcast problem in the telephone model, thus proving an $O(\frac{\log n}{\log \log n})$ approximation (Section 2.3.2).
4. We show that the case of edge-star symmetric multicommodity multicast problem has the same optimal solution (up to poly-log factors) as telephone multicommodity multicast, yielding a $2^{O(\log \log n \sqrt{\log n})}$ approximation (Section 2.3.3).
5. We give an $O(n^{\frac{2}{3}})$-approximation for the general (asymmetric) multicommodity multicast problem in the edge-star model (Section 2.3.4).

Table 2.1 contains a summary of our results in context.

	Broadcast	Gossip	Multicommodity
Radio	$D + O(\log^2 n)$ [59]	$O(D + \Delta \log n)$ [37]	Unknown
		$\Omega(n^{1/2-\epsilon})$ hard*	$\Omega(n^{1/2-\epsilon})$ hard*
Edge-star	OPT$= D$	OPT$\cdot O(\frac{\log n}{\log \log n})$*	OPT$\cdot \tilde{O}(2^{\sqrt{\log n}})$*(symmetric)
			OPT$\cdot O(n^{\frac{2}{3}})$* (asymmetric)
Telephone	OPT$\cdot O(\frac{\log n}{\log \log n})$ [21]	OPT$\cdot O(\frac{\log n}{\log \log n})$ [21]	OPT$\cdot \tilde{O}(2^{\sqrt{\log n}})$ [66]

Table 2.1: A summary of upper and lower bounds achieved in the different problems. We prove the results marked * in the table.

2.2 Lower bound for gossip in the radio model

In this section, we show it is NP-hard to approximate gossip in the radio model within a factor of $O(n^{1/2-\epsilon})$. This also implies the same hardness result for multicommodity multicast under the radio model, because gossip is a special case of multicommodity multicast. In order to show these hardness results, we first consider the smallest set of induced matchings which cover the vertices of a bipartite graph.

Definition 2.2.1. An *induced matching* is a matching of some vertices U in a graph G, such that $G[U]$ is a matching. (We use $G[U]$ to mean the graph G induced on the vertex set U.) In other words, in the graph G only the matching edges are present between the nodes in U.

A *covering set of induced matchings* (CSIM) is a set of induced matchings which cover all the vertices in the graph. The size of a covering set of induced matchings is defined to be the number of induced matchings.

First, we will show the hardness of finding a minimum CSIM by a reduction from coloring. Then we will use the hardness of minimum sized CSIM to prove the hardness results for radio gossip.

Theorem 2.2.2. It is NP-hard to approximate CSIM to within a $n^{1/2-\epsilon}$ factor for any constant $\epsilon > 0$.

Proof. Given a coloring instance $G = (V, E)$, we first turn this into a bipartite graph, where we want to find a CSIM. For each $v \in V$ we make $n+1$ copies of v in each side of the partition; $v_1^L, v_2^L, \ldots v_{n+1}^L$ for L and $v_1^R, v_2^R, \ldots v_{n+1}^R$ for R. We use the edges $E_v = \{(v_i^L, v_i^R) | v \in V, i \in [n+1]\}$, called the straight edges and $E_e = \{(u_i^L, v_j^R) | uv \in E, i, j \in [n+1]\}$, called the cross edges. Now $G' = (L, R, E_v \cup E_e)$ is the bipartite graph for which we want to find a CSIM. Figure 2.1 shows an example construction.

Let χ be the number of colors in an optimal coloring in G. Let λ be the number of sets in a minimal CSIM in G'.

We now show that $\lambda \leq \chi \leq n$. Let C_i be a set of vertices of color i in the coloring on G. If we take the edges $\{(v_j^L, v_j^R) | v \in C_i, j \in [n+1]\}$, they are an induced matching. Each vertex has one straight edge in G', and if a vertex is used in the matching, then its straight edge is used. So, we only need to show that no cross edges go between vertices in this matching. If a cross edge (u_j^L, v_k^R) did exist, then $(u, v) \in E$ but then u, v couldn't be the same color. So, for each color we have defined an induced matching. These induced matchings cover all the nodes since every node receives some color in the coloring on G.

Now we will show that $\chi \leq \lambda$ or $n+1 \leq \lambda$. Let $S_1, S_2, \ldots S_\lambda$ be the induced matchings covering G'. Assume that there is some $v \in V$ that has all of its corresponding vertices in G' matched via cross edges. Then we can only have at most one cross edge per induced matching adjacent to the v_i^L's. If an induced matching has (v_i^L, u_ℓ^R) and $(v_j^L, w_{\ell'}^R)$ then this is not an induced matching since (v_j^L, u_ℓ^R) is an edge. Therefore in this case to match all the v_i^L in some induced matching, we will need at least $n+1$ induced matchings. Now consider each $v \in G$ has one of its straight edges used in some induced matching. Let S_j be the first induced matching containing a straight edge adjacent to some v_i^L. In S_j, because some v_i^L is matched via its straight edge, then no

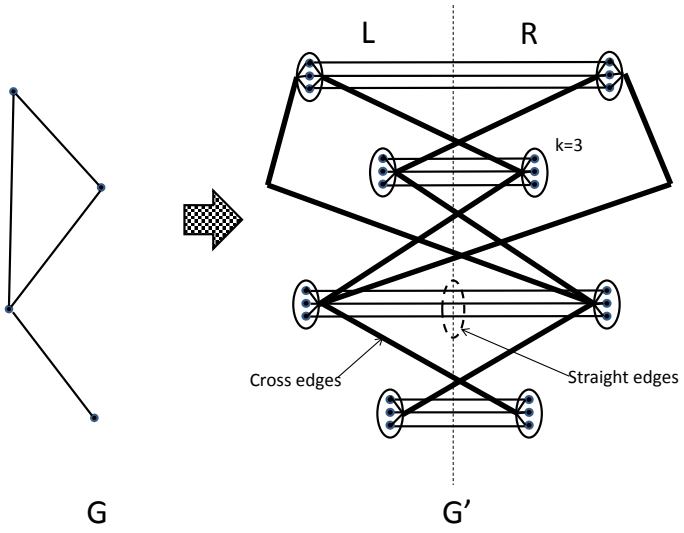


Figure 2.1: Here is an example of the construction of G' from G. The thick edges represent complete bipartite subgraphs.

v_ℓ^L is matched via a cross edge. Now in G color v with the jth color. This is a valid coloring. If some (v_i^L, v_i^R) and (u_ℓ^L, u_ℓ^R) were both in the same induced matching, then there can't be the edge (u, v) in the original graph G.

Combining the above two parts we get that $\chi = \lambda$.

We begin with a graph G such that it is NP-hard to distinguish if there is a coloring of size $|V(G)|^\epsilon$ from if the coloring requires at least $|V(G)|^{1-\epsilon}$ colors [26]. Therefore, in the graph G' we created, it is NP-hard to distinguish if there is a set of induced matchings that cover the vertices of size n^ϵ or n. We have $O(n^2)$ vertices in G' though. So, in a graph with n vertices it is NP-hard to approximate the number of induced matchings needed to cover the vertices within a factor of $O(n^{1/2-\epsilon})$. □

Now that we have developed the hardness result for CSIM, we will use the graph we created for CSIM, to create instances of radio gossip.

Corollary 2.2.3. It is NP-hard to approximate radio gossip to within a $n^{1/2-\epsilon}$ factor for any constant $\epsilon > 0$.

Proof. We convert the induced matching instance to a gossip problem in a similar fashion to above. We can consider that we have the bipartite graph G' and we build a complete binary tree with its leaves being the nodes v_i^L. The terminal nodes in the gossip problem are set to be all the nodes. To communicate the message to all other nodes, each node v_i^R must at some point be the only node trying to talk to some node on the other side of the bipartition. In other words, we need to have induced matchings at each point in order for the v_i^R to propagate their messages to some other node without interference. Therefore, we need at least as many induced matchings as it takes to cover the graph to complete the gossip. Call this number C; we can now achieve gossip in

time $2C + 3\log n$ as follows. We do this by using the induced matchings so that each vertex v_i^R communicates its message to someone on the other side of the partition. Next we propagate the message up the binary tree to the root node. This takes time at most $2\log n$ since at each node of the path in the binary tree, a message can be delayed only for two steps, and the path length is logarithmic. Then we broadcast the message down the tree. This takes time $\log n$ since we can use the edge-star model to just broadcast all the gathered messages from the root along the down-stars in one time step per level. Lastly, we need to communicate the message back to the v_i^R, which takes time C. We know that radio gossip takes time at least C and can be done in time $2C + 3\log n$ on this graph.

Therefore, it is NP-hard to approximate radio gossip better than a factor of $O(n^{1/2-\epsilon})$ otherwise, we could approximate the CSIM within the same factor. □

2.3 The Edge-Star Model

In this section, we consider the edge-star model which generalizes the telephone model. We focus on three specific classes of problems; gossip, symmetric multicommodity multicast, and asymmetric multicommodity. In the *symmetric multicommodity* problem, we are given a set of demand pairs, and if (s_i, t_i) is a demand, then (t_i, s_i) is also a demand. In the asymmetric multicommodity case, there are no restrictions on which demand pairs are present.

In Section 2.3.1, we first obtain an $O(\frac{\log n}{\log\log n})$ approximation algorithm for gossip in the edge-star model by reducing the problem to the broadcast problem in the telephone model. Next, in Section 2.3.2, we consider the special case where the underlying graph is a tree. In this special case, then we show that the multicommodity multicast in the edge-star model reduces to the broadcast problem in the telephone model, yielding an $O(\frac{\log n}{\log\log n})$ approximation. In Section 2.3.3, we show that the case of edge-star symmetric multicommodity multicast problem has the same optimal solution (up to poly-log factors) as telephone multicommodity multicast, yielding an $\tilde{O}(2^{\sqrt{\log n}})$ approximation. Lastly, in Section 2.3.4, we give an $O(n^{\frac{2}{3}})$-approximation for the general (asymmetric) multicommodity multicast problem in the edge-star model.

2.3.1 Gossip

Here we show an $O(\frac{\log n}{\log\log n})$ approximation for edge-star gossip. First, we show that a solution to the gossip problem in the edge-star model gives a solution to the broadcast problem in the telephone model of the same length. Next, we show that using a solution for the broadcast problem in telephone we can get a solution of twice the length to the gossip problem in the edge-star model. This show that their optimal solutions differ in cost by a factor of at most two.

Lemma 2.3.1. The optimal broadcast time in the telephone model is no more than the optimal gossip time in the edge-star model.

Proof. Let $\mathcal{S}$ denote an optimal schedule for gossip in the edge-star model that completes in T rounds. Let r denote the root node for the broadcast problem in the telephone model. Fix a node v. Let P_v denote a path taken by the message from v to arrive at r in the schedule S. Let E_t denote

the set of all directed edges in $\cup_v P_v$ that are activated in round t in S. By definition of the edge-star model, if (u_1, v_1) and (u_2, v_2) are in E_t, then $v_1 \neq v_2$. Furthermore, by our choice of the paths, we obtain that (i) for any distinct (u_1, v_1) and (u_2, v_2) in E_t, $u_1 \neq u_2$; and (ii) the edges of P_v appear in order of increasing time in the collection of E_ts.

We now argue that a reverse schedule in which the activated sets are given by $E'_t = \text{REV}(E_{T-t})$ forms a broadcast schedule from the root, where $\text{REV}(X)$ equals $\{(v, u) : (u, v) \in X\}$ for any set X of directed edges. In any round t, for any distinct (u_1, v_1) and (u_2, v_2) in E_t, we have $u_1 \neq u_2$ and $v_1 \neq v_2$; therefore, $\text{REV}(E_t)$ is a matching. Since the edges of P_v appear in order of increasing time in the collection of E_ts, the edges of the $\text{REV}(P_t)$ appear in order of increasing time in the collection of E'_ts. Consequently, the message from the root is delivered to each node in T rounds. □

Lemma 2.3.2. The optimal gossip time in the edge-star model is no more than twice the optimal broadcast time in the telephone model.

Proof. The proof mirrors the proof of Lemma 2.3.1. Let $\mathcal{S}$ denote an optimal schedule for broadcast from root r in the telephone model that completes in T rounds. Fix a node v. Let P_v denote a path taken by the message from r to arrive at v in the schedule S. Let E_t denote the set of all directed edges in $\cup_v P_v$ that are activated in round t in S. By definition of the telephone model, for distinct (u_1, v_1) and (u_2, v_2) in E_t, $u_1 \neq u_2$ and $v_1 \neq v_2$. Furthermore, by our choice of the paths, we obtain that the edges of P_v appear in order of increasing time in the collection of E_ts.

We now argue that a reverse schedule in which the activated sets are given by $E'_t = \text{REV}(E_{T-t})$ forms a schedule for gathering in the edge-star model. In any round t, for any distinct (u_1, v_1) and (u_2, v_2) in E_t, we have $u_1 \neq u_2$ and $v_1 \neq v_2$; therefore, $\text{REV}(E_t)$ is a matching, and is a valid set of edges to activate in the edge-star model in round $T - t$. Since the edges of P_v appear in order of increasing time in the collection of E_ts, the edges of the $\text{REV}(P_t)$ appear in order of increasing time in the collection of E'_ts. Consequently, the message from any node v is delivered to the root in T rounds.

Once the root has all the messages, we can complete the gossip by running the broadcast schedule. Since any schedule in the telephone model is valid in the edge-star model, it follows that this broadcast completes in T rounds. We thus have a gossip schedule that completes in the edge-star model in $2T$ rounds. □

There exists an $O(\frac{\log n}{\log \log n})$ approximation for telephone broadcast [21]. Therefore this same approximation holds for the edge-star gossip problem.

2.3.2 Multicommodity multicast on a tree

In this part, we consider the multicommodity multicast problem in the edge-star model in the special case where our host graph is a tree. Here we give a reduction to telephone broadcast. When the host graph is a tree, the path taken by any message is known, so we simply need to coordinate the communications.

Lemma 2.3.3. There is an $O(\frac{\log n}{\log \log n})$ approximation for the edge-star multicommodity multicast problem in a tree.

Proof. We will start by choosing some vertex r to be the root of the tree. Let the optimal solution take time D (we can try all $2n$ possible values for D only losing a polynomial factor in runtime). Now for each demand pair, (s_i, t_i) the message will have to go from s_i to $lca(s_i, t_i)$, and then from the $lca(s_i, t_i)$ down to t_i. Bringing all the messages down the tree from $lca(s_i, t_i)$ to t_i can be done in time $D + 1$; we spend $D + 1$ time steps alternating between the odd layers broadcasting their messages down and the even layers broadcasting their message down. Since each layer is a collection of edge-disjoint stars, this can be implemented in one round in the edge-star model.

The hard part is bringing the messages up from s_i to $t'_i = lca(s_i, t_i)$. So, we will consider that we simply have the constraints of the form (s_i, t'_i). First we will break the tree up into sets of $2D$ consecutive layers starting every D layers. This guarantees that every constraint (s_i, t_i) is in some set of $2D$ layers.

Now consider some $2D$ layers in the tree. Look at the union of all the (s_i, t'_i) paths in these layers. These form a forest, where each tree has depth at most $2D$ and each node has a max degree of D; so each tree has poise $5D$ (recall that the poise is the sum of the maximum degree and the diameter). Therefore each of these trees can gather all their messages to their uppermost nodes in time $O(D\frac{\log n}{\log \log n})$.

We can run all the gathers to satisfy (s_i, t'_i) in two groups; we can run every other set of $2D$ layers in the tree simultaneously, as they are disjoint. Hence, in time $O(D\frac{\log n}{\log \log n})$, we can satisfy the demands (s_i, t'_i). After this, in $D + 1$ more steps, we can satisfy the demands (t'_i, t_i). Therefore in time $O(D\frac{\log n}{\log \log n})$ we satisfy all the (s_i, t_i) demands. □

2.3.3 Symmetric Multicommmodity Multicast

Note that the symmetric multicommodity multicast problem in the telephone model is equivalent (within constant factors) to the general multicommodity multicast problem [7, 70] for which an $\tilde{O}(2^{\sqrt{\log k}})$ approximation algorithm is known, where k is the number of terminals [66]. We show a reduction from the symmetric multicommodity multicast problem in the edge-star model to the symmetric multicommodity multicast problem in the telephone model, losing an additional $O(\frac{\log^3 n}{\log \log n})$ factor in the approximation ratio in an n-node graph.

Theorem 2.3.4. Given a ρ-approximation for the symmetric multicommodity multicast problem on k terminal pairs in an n-node undirected graph under the telephone model, we can design an $O(\rho \cdot \log^2 k \cdot \frac{\log n}{\log \log n})$ approximation for the same problem in the edge-star model.

Proof. Given an optimal solution to symmetric multicommodity multicast in the edge-star model, we demonstrate a solution to the symmetric multicommodity multicast problem in the telephone model with a poly-log multiplicative loss in performance. Consider an input instance with demand pairs $\{s_i, t_i\}$ for $i = 1 \cdots k$ on an undirected graph G. Consider an optimal schedule for the edge-star symmetric multicommodity multicast problem on this instance. This defines for each pair s, t, a pair of paths from one node to the other where the edges of the paths are labeled in increasing time order denoting the periods in which these edges participated in an information transmission. Suppose the optimal time for multicasting is L; then these paths are of length at most L. Also, given the in-degree one bound for the edge-star model (each receiver can listen to at most one transmitter

in this wireless model), the indegree of the sugraph representing the union of these optimal transmissions is also at most L. Our goal is to use these paths to aggregate the messages from a set of these pairs into a subset of carefully selected terminals using a reverse broadcast scheme, and then transmit the aggregated messages back to the corresponding mates of these sources. Both these steps of gathering and sending will be accomplished using multicommodity multicast instances in the telephone model.

To define the aggregation pattern, define an auxiliary graph H with one node per *demand pair* s_i, t_i. This graph is only for the sake of argument so we will use optimal paths in the edge-star multicommodity multicast scheme in defining it. Note that the optimal transmission paths for a pair represent two paths: one from s_i to t_i and the second from t_i to s_i, where these two paths may share edges. Concatenated together they define what we will call an "optimal cycle" for this pair. Define an edge between two pairs if their optimal cycles intersect at a node. In Figure 2.2, we can see an example of when optimal cycles intersect. Thus H defines the conflict or interference between the demand pairs in the optimal multicommodity multicast schedule in the edge-star model.

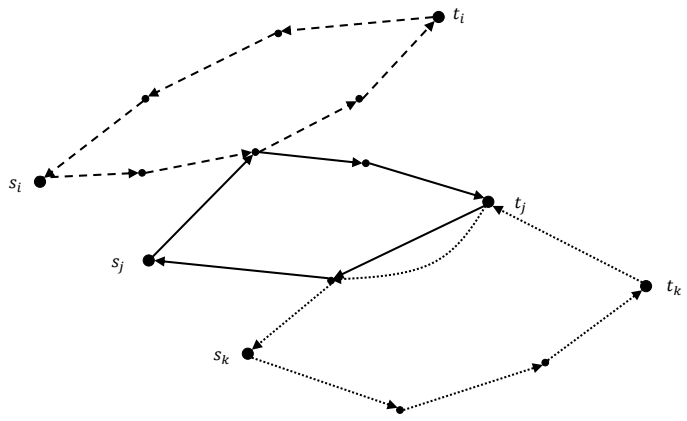


Figure 2.2: Here is an example of the optimal paths between some (s_i, t_i) pairs. Here we see that the (s_j, t_j) pair intersects (s_i, t_i) and (s_k, t_k), but (s_i, t_i) and (s_k, t_k) do not intersect.

We now use a network decomposition procedure [5] on H to decompose the k demand pairs into $\log_2 k$ disjoint layers with the following property: the set of nodes in each layer can be decomposed into node-disjoint shallow trees, i.e., each tree in one of the layers has diameter at most $2 \log_2 k$. This decomposition is done as follows: pick any vertex v in H and build a BFS tree from v. Now let i be the smallest depth such that the number of nodes at depth i or less is more than the number of nodes at depth $i + 1$. Put v and everything within distance i of v into the current layer. Now remove v and it's BFS tree up to depth $i + 1$ from H. Repeat this process to form each layer. Once H is empty, let U be the vertices not yet assigned to a layer. Then start forming a new layer from the graph $H[U]$.

This process assigns at least half of the remaining nodes to the current layer, hence we build at

most $\log_2 k$ layers. The diameter of each component in a layer is at most $2\log_2 k$, because as we move down the BFS tree the number of nodes contained in it double at each step.

Now we can use these layers to define our gathering problems. Consider one layer i and one tree $T_{i,j}$ in this layer in the decomposition. This represents a shallow subgraph in H, so let us root this at a demand pair denoted P_{ij}. By following the paths in this subgraph from every other node to P_{ij}, we can replace their intersections with paths in the optimal multicast originating at each terminal s in any of the pairs to one of the two terminals, say t_{ij} in the pair P_{ij}. This defines one of the gathering trees gathering to the terminal t_{ij}. By construction, the in-degree of any node in the gathering tree is at most L and the distance from any node to the root t_{ij} is at most $O(L\log k)$. Note that by the disjointness of the subgraphs in one layer i, all the gather trees are node disjoint. For each gather tree T_{ij}, we now set up a gathering multicast problem with all the terminals in the tree going to the root t_{ij}. Note that since each tree has total degree + diameter at most $O(L\log k)$, the poise of each tree is bounded by $O(L\log k)$ and thus each of these trees has a gathering schedule in the telephone model taking at most $O(Poise\cdot\frac{\log n}{\log\log n})$ steps in an n-node graph [70]. This gives a feasible solution to the set of all gathering problems in one layer i running in time $O(L\cdot\log k\cdot\frac{\log n}{\log\log n})$. Repeating this over the layers finally gives a set of gathering problems in the telephone model that complete in total time $O(L\cdot\log^2 k\cdot\frac{\log n}{\log\log n})$.

Note that the same schedules can be reversed to send all the gathered information in each tree to all the terminals in a tree finishing the requirements. Employing a ρ-approximation for this multicommodity multicast problem in the telephone model proves the theorem. □

2.3.4 Asymmetric Multicommodity Multicast

For the edge-star asymmetric multicommodity multicast problem, we will use the network decomposition used in the previous proof, along with telephone broadcast in trees with small poise.

Theorem 2.3.5. There is an $\tilde{O}(n^{\frac{2}{3}})$-approximation for the asymmetric multicommodity multicast problem in the edge-star model.

Proof. We develop the algorithm in two phases. First, we design an $\tilde{O}(\sqrt{p})$-approximation algorithm for the case with p demand pairs (note that p can be up to $O(n^2)$ in an n-node graph). Then we combine this with an algorithm that satisfies all the demands in the in-neighborhood of a node in the demand graph with high indegree to get the final result.

A Greedy Algorithm. To design the $\tilde{O}(\sqrt{p})$-approximation algorithm, we use a greedy method: assume that the value of the optimal multicast time is L (we can try all the $2n$ possible guesses in parallel to dispense this assumption with a polynomial running-time overhead). For every unsatisfied demand pair (s_i, t_i) (note that demand pairs are ordered in the asymmetric case), we look for a path of length at most L from s_i to t_i. If we find one, we add it to the greedy collection and delete all the nodes in this path. Suppose we are able to collect g paths for the pairs denoted G in the greedy phase until we can find no more paths of small length for the remaining demands.

Now it must be the case that all optimal paths for the remaining demands in $P\setminus G$ must intersect the greedy paths. This implies that for every demand pair (s,t) in $P\setminus G$, we can follow its optimal path to its intersection with one of the greedy paths, say for the pair (s_i,t_i), and then continue in the greedy path to t_i. In this way, every demand source in $P\setminus G$ can be routed and assigned to one

of the sinks in the greedy pairs G in a collection of paths: each such path has length at most $2L$ (coming from at most L steps to the intersection with the greedy collection and another L from the intersection to the sink at the end of this greedy path); also the indegree of the collection of these paths is at most $L+1$ since they arise from the optimal collection plus the greedy subgraph which adds at most one to each node's indegree. We now set up a dummy broadcast problem (following Nikzad and Ravi [66]) by hooking up the set of sinks at the end of the greedy paths, say $T(G)$, as leaves in a complete binary tree with new dummy nodes and a dummy root t. We solve for the broadcast problem in this graph from the dummy root t to all the sources s_i in all the pairs. By the above construction, there exists a tree of poise $O(L+\log n)$ that connects all the sources to this root. From the result of Ravi [70], this implies a broadcast scheme completing in $O(L\frac{\log n}{\log\log n})$. Using an α-approximation algorithm for broadcast in the telephone model, we get a tree that assign the sources in P to the sinks in $T(G)$ in $O(\alpha \cdot L\frac{\log n}{\log\log n})$ steps. Let us denote the set of sinks in $T(G)$ by $t'_1, \cdot, t'_g$ and the set of sources assigned to a sink t'_i by S_i.

The remaining task is to send back the messages gathered from S_i at t'_i to the sinks corresponding to the sources in S_i - let us denote this sink set by T_i. Note that by construction, all the sinks in T_i are at a distance at most $O(\alpha \cdot L\frac{\log n}{\log\log n})$ from t'_i by following the paths to the corresponding source s and then concatenating the undirected path to its mate t. However, these local broadcasts must obey the edge-subgraph condition of having indegree at most one which is tricky to enforce.

If the number of greedy pairs $g = |G|$ is at least $\sqrt{p}$, we simply satisfy these pairs and move to the next iteration: the number of such iterations is at most $\sqrt{p}$ and each iteration can be implemented in $O(L)$ steps (running the disjoint greedy path schedules in parallel). If the number of pair is less than $\sqrt{p}$, we can carry out the broadcast from each greedy sink t'_i to its sink set T_i in time $O(\alpha \cdot L\frac{\log n}{\log\log n})$ by reversing the gathering in the earlier broadcast tree and extending it to the corresponding sinks. Processing these trees one after another, we use a total of $O(\sqrt{p}\cdot\alpha\cdot L\frac{\log n}{\log\log n})$. Since α is sublogarithmic [24], we finally get an $\tilde{O}(\sqrt{p})$-approximation as claimed.

A Local Algorithm. For the second ingredient we observe that if the in-degree of any node v in the demand graph is δ, then we can satisfy all the demand requirements of the predecessors of v in the demand graph $In(v)$ in time $\tilde{O}(L)$. Note that since all the terminals in $In(v)$ send their message to v, the union of the directed paths that transmit these messages in the optimal solution have distance at most L from the terminals to v and induce an in-degree of at most L. This defines a tree of poise $O(L)$ and hence enables us to find a broadcast scheme that gathers all the messages from $In(v)$ at v in time $\tilde{O}(L)$. By reversing this broadcast tree and then following the optimal paths from each terminal in $In(v)$ to its other sinks, we can find a tree of depth (not poise) at most $\tilde{O}(L)$ rooted at v where these messages are gathered. Since v is the only node sending out the gathered messages, we can send all these messages to their intended sinks in a breadth-first tree in time $\tilde{O}(L)$ in the edge-star model. Note that we have taken care of all the demands originating in $|In(v)|$ nodes.

Combining the two algorithms. We can now combine the two algorithms as follows: As long as p, the number of demand pairs in the n-node graph, is at least $\Omega(n^{\frac{4}{3}})$, we use the local algorithm. By averaging over the indegrees that partition the demand pairs, there exists a node of indegree at least $\Omega(n^{\frac{1}{3}})$ in the demand graph. The local algorithm thus satisfies the demands originating in at least this many nodes in one iteration. The number of iterations is thus at most $n^{\frac{2}{3}}$ each taking

$\tilde{O}(L)$ multicast steps. On the other hand, when p drops below $O(n^{\frac{4}{3}})$, we use the greedy algorithm to get an approximation ratio of $\tilde{O}(\sqrt{p}) = \tilde{O}(n^{\frac{2}{3}})$ giving the result. □

2.4 Conclusion

We have obtained new results in the approximability of rumor spreading problems in the well-studied radio model as well as a new model motivated by wireless communications, which we call the edge-star model. For the radio model, we present an $\Omega(n^{1/2-\epsilon})$ hardness of approximation bound for radio gossip, making progress on an open problem mentioned in [36]. For the edge-star model, we present an $O(\frac{\log n}{\log \log n})$ approximation algorithm for gossip, an $\tilde{O}(2^{\sqrt{\log n}})$ approximation algorithm for symmetric multicommodity multicast, and an $\tilde{O}(n^{2/3})$ approximation algorithm for asymmetric multicommodity multicast. Our approximation algorithms expose relationships between the edge-star model and the well-studied telephone model.

Our work leaves several interesting open problems. Among the nine cells listed in the matrix of Table 2.1 of Section 2.1, only radio broadcast and edge-star broadcast are resolved. Significant gaps between the best known upper and lower bounds on approximability remain for telephone broadcast, the gossip problem under all three models, and the multicommodity multicast problem under all three models. In the edge-star model, the symmetric and asymmetric versions of the multicommodity multicast problem are distinct, and both are open, in terms of the best approximation factor achievable in polynomial-time.

Chapter 3

Planar Gossip

3.1 Introduction

Rumor spreading in networks has been an active research area with questions ranging from finding the minimum possible number of messages to spread gossip around the network [6, 44, 80] to finding graphs with minimum number of edges that are able to spread rumors in the minimum possible time in the network [42]. There is also considerable work in the distributed computing literature on protocols for rumor spreading and gossip based on simple push and pull paradigms (e.g., see [27, 29, 38, 51]).

The focus of this chapter is the class of problems seeking to minimize the time to complete the rumor spread, the prototypical example being the **minimum broadcast time problem** where a message at a root node must be sent to all nodes via connections represented by an undirected graph in the minimum number of rounds. Under the popular "telephone" model, every node can participate in a telephone call with at most one other neighbor in each round to transmit the message, and the goal is to minimize the number of rounds. This problem has seen active work in designing approximation algorithms [8, 24, 57, 70]. One generalization of broadcast is the **minimum multicast time problem**: We are given an undirected graph $G(V, E)$ representing a telephone network on V, where two adjacent nodes can place a telephone call to each other. We are given a source vertex r and a set of terminals $R \subseteq V$. The source vertex has a message and it wants to inform all the terminals of the message. To do this, the vertices of the graph can communicate in rounds using the telephone model. The goal is to deliver the message to all terminals in the minimum number of rounds.

Recently, a more general demand model called the multicommodity multicast was introduced in [66]. In the **minimum multicommodity multicast time problem**, a graph $G(V, E)$ is given along with a set of pairs of nodes $P = \{(s_i, t_i) | 1 \leq i \leq k\}$, known as demand pairs. Each vertex s_i has a message m_i which needs be delivered to t_i. The vertices communicate similar to the multicast problem. The goal is to deliver the message from each source to its corresponding sink in the minimum number of rounds. Note that there is no bound on the number of messages that can be exchanged in a telephone call. In this sense, the telephone model captures a classic information dissemination problem where the primary communication constraint is the number of connections

that a given node can make in each round, not link bandwidth.

3.1.1 Poly-logarithmic approximation for planar multicommodity multicast.

While even sub-logarithmic ratio approximations have been known for the minimum time multicast problem [8, 24, 57, 70], the best known approximation guarantees for the multicommodity case [66] is $\tilde{O}(2^{\sqrt{\log k}})$ where k is the number of different source-sink pairs.

Theorem 3.1.1. There is a polynomial time algorithm for minimum time multicommodity multicast with k source-sink pairs in n-node undirected planar graphs that constructs a schedule of length $O(OPT \log^3 k \frac{\log n}{\log \log n})$ where OPT is the length of the optimal schedule.

This result to extends in a natural way to bounded genus graphs. Our results make critical use of the fact that planar graphs admit small-size balanced vertex separators that are a combination of three shortest paths starting from any given node [79]. We aggregate messages at the paths, move them along the path and then move them onto their destinations using a local multicast. To break the overall multi-commodity multicast problem into recursive subproblems, we solve an LP relaxation for the overall problem and for those pairs for which the LP uses the separator path nodes in sending messages by a "large" amount, we aggregate them to the separator paths and move them along the paths. However, to define this aggregation automatically we need to use a linear program which requires us to relate another lower bound for the schedule length that we describe next.

3.1.2 Poise and a new LP rounding algorithm.

Suppose that the (single-commodity) multicast problem in a graph G with root r and terminals R admits a multicast schedule of length L. Consider all the nodes $I \subseteq V$ in the graph that are informed of the message from the root in the course of the schedule. For every node $v \in I$ consider the edge through which v first heard the message and direct this edge into v. It is easy to verify that this set of arcs forms an out-arborescence T rooted at r and spanning I. In particular, every node in I except r has in-degree exactly one and there is a directed path from r to every vertex in I.

Definition 3.1.2. Define the poise of an undirected tree T to be the sum of the diameter of the tree and the maximum degree of any node in it. Define the poise of a directed tree to be that of its undirected version (ignoring directions).

The discussion above of constructing a directed tree from a multicast schedule implies that the poise of the tree constructed from a multicast schedule of length L is at most 3L (see also [70]). The following lemma gives the relation in the other direction.

Lemma 3.1.3. [70] Given a tree on n nodes of poise L, there is a polynomial time algorithm to construct a broadcast scheme of length $O(L \cdot \frac{\log n}{\log \log n})$ from any root.

Note that a complete d-ary regular tree of depth d requires time d^2 to finish multicast from the root; If the size of the tree is n, then $d = O(\frac{\log n}{\log \log n})$. For this tree $L = O(\frac{\log n}{\log \log n})$ while any broadcast scheme takes $\Omega((\frac{\log n}{\log \log n})^2)$ steps showing that the multiplicative factor is necessary.

Even though approximation algorithms for minimum poise trees connecting a root to a set of terminals were known from earlier work [8, 24, 70], their guarantees are with respect to an optimal (integral) solution and not any specific LP relaxation. In particular, the LP-based algorithm of [70] rounds a solution to the poise LP in phases without preserving the relation of the residual LPs that arise in the phases to the LP for the poise of the whole graph. Similarly, the LP-based algorithm of [8] solves a series of LPs determining how to hierarchically pair terminals and form the desired broadcast tree with cost within a logarithmic factor of the integral optimum poise, but without relating the resulting tree to the LP value of the poise of the original graph. It is not straightforward to use these methods to derive an integrality gap for the minimum poise LP, and this has remained an open problem. Deriving an approximation algorithm for minimum poise subgraphs for the single-commodity multicast version with a small *integrality gap* is a critical ingredient in our approximation algorithm for multicommodity multicast problem in planar graphs (Theorem 3.1.1). We derive the first such result.

Theorem 3.1.4. Given a fractional feasible solution of value L to a natural linear programming relaxation of the minimum poise of a tree connecting a root r to terminals R (POISE-L LP, see Section 3.2), there is a polynomial time algorithm to construct a tree spanning $r \cup R$ of poise $O(L \log k)$ where $k = |R|$ and $n = |V|$.

Our LP rounding for minimum poise are based on exploiting a connection to the theory of multiflows [15, 30, 63]; this is an interesting technique in its own right that we hope will be useful in obtaining other LP rounding results for connectivity structures while preserving degrees and distances.

3.1.3 Radio Gossip in Planar Graphs.

Our techniques for addressing multicommodity multicast are also applicable to radio gossip in planar graphs. In the radio model of communication that also occurs in rounds, a transmitting node may broadcast to multiple nodes in around but a node may receive successfully in a given time step only if exactly one of its neighbors transmits. The gossip problem is a special case of the multicommodity multicast problem where the demand pairs include all possible pairs of nodes (alternately, every node's message must be transmitted to every other node). The minimum gossip problem in the radio model has been widely studied [37] but all known upper bounds involve both the diameter and degree of the network. In particular, for general n-node graphs, there is an $\Omega(n^{\frac{1}{2}-\epsilon})$-hardness of approximation result for computing a minimum gossip schedule [47]. Our next result breaks this barrier for planar graphs (the proof and algorithm are in Section 3.4).

Theorem 3.1.5. There is a polynomial time algorithm for minimum time radio gossip in an n-node undirected planar graph that constructs a schedule of length $O(OPT \cdot \log^2 n)$ where OPT is the length of the optimal gossip schedule.

Since radio broadcast from any node can already be achieved with additive poly-logarithmic time overhead above the optimum [64], our algorithm for radio gossip focuses on gathering all the messages to a single node. For this, we use the path-separator decomposition in planar graphs to recursively decompose the graph and gather messages bottom up. However, the diameter of subgraphs formed by the decomposition are not guaranteed to be bounded so we use a carefully

constructed degree-bounded matching subproblem to accomplish the recursive gathering: these techniques adapt and extend the methods used for constructing telephone multicast schedules [66] but apply them for the first time to the radio gathering case.

3.1.4 Minor-free Graphs

In Section 3.5, we show that both our results on planar graphs also naturally extend to minor-free graphs, as similar path separator results are also known for minor-free graphs [1].

Theorem 3.1.6. There is a polynomial time algorithm for minimum time multicommodity multicast with k source-sink pairs in n-node undirected H-minor-free graph for a constant sized H that constructs a schedule of length $O(OPT \log^3 k \frac{\log n}{\log \log n})$ where OPT is the length of the optimal schedule.

Theorem 3.1.7. There is a polynomial time algorithm for minimum time radio gossip in an n-node undirected H-minor-free graph for a constant sized H that constructs a schedule of length $O(OPT \log^2 n)$ where OPT is the length of the optimal gossip schedule.

3.1.5 Previous Work

Minimum time multicast in the telephone model. Finding optimal broadcast schedules for trees was one of the first theoretical problems in this setting and was solved using dynamic programming [69]. For general graphs, Kortsarz and Peleg [57] developed an additive approximation algorithm which uses at most $c \cdot OPT + O(\sqrt{n})$ rounds for some constant c in an n-node graph. They also present algorithms for graphs with small balanced vertex separators with approximation ratio $O(\log n \cdot S(n))$ where $S(n)$ is the size of the minimum balanced separator on graphs of size n from the class. The first poly-logarithmic approximation for minimum broadcast time was achieved by Ravi [70] and the current best known approximation ratio is $O(\frac{\log n}{\log \log n})$ due to Elkin and Korsartz [24]. The best known lower bound on the approximation ratio for telephone broadcast is $3 - \epsilon$ [22].

In his study of the telephone broadcast problem, Ravi [70] introduced the idea of finding low poise spanning trees to accomplish broadcast. In the course of deriving a poly-logarithmic approximation, Ravi also showed how a tree of poise P in an n-node graph can be used to complete broadcast starting from any node in $O(P \cdot \frac{\log n}{\log \log n})$ steps. His result provided an approximation guarantee with respect to the optimal poise of a tree but not its natural LP relaxation that we investigate.

Guha et al. [8] improved the approximation factor for multicasting in general graphs to $O(\log k)$ where k is the number of terminals. The best known approximation factor for the multicast problem is $O(\frac{\log k}{\log \log k})$ [24]. Both of [8, 24] present a recursive algorithm which reduces the total number of uninformed terminals in each step of the recursion, while using $O(OPT)$ number of rounds in that step. In [8], they reduce the number of uninformed terminals by a constant factor in each step and so they obtain a $O(\log k)$-approximation, but in [24], the number of uninformed terminals is reduced by a factor of OPT which gives a $O(\frac{\log k}{\log \log k})$-approximation due to the fact that $OPT = \Omega(\log k)$. These papers also imply an approximation algorithm with factors $O(\log k)$ and

$O(\frac{\log k}{\log \log k})$ for the Steiner minimum poise subgraph problem; however, these guarantees are again with respect to the optimum integral value for this problem and not any fractional relaxation.

For the multicommodity multicast problem, Nikzad and Ravi [66] adapt the methods of [23, 24] to present an algorithm with approximation ratio $\tilde{O}(2^{\sqrt{\log k}})$ where k is the number of different source-sink pairs. They also show that there is a poly-logarithmic approximation inter-reducibility between the problem of finding a minimum multicommodity multicast schedule and that of finding a subgraph of minimum generalized Steiner poise (i.e., a subgraph that connect all source-sink pairs, but is not necessarily connected overall, and has minimum sum of maximum degree and maximum distance in the subgraph between any source-sink pair).

Radio Gossip. The radio broadcast and gossip problems have been extensively studied (see the work reviewed in the survey [36]). The best-known scheme for radio broadcast is by Kowalski and Pelc [59] which completes in time $O(D + \log^2 n)$, where n is the number of nodes, and D is the diameter of the graph and is a lower bound to get the message across the graph from any root. The $O(\log^2 n)$ term is also unavoidable as demonstrated by Alon et al. [3] in an example with constant diameter that takes $\Omega(\log^2 n)$ rounds for an optimal broadcast scheme to complete. Elkin and Korsartz [20] also show that achieving a bound better than additive log-squared is not possible unless $NP \subseteq DTIME(n^{\log \log n})$. For planar graphs, the best upper bound for radio broadcast time is $D + O(\log n)$ given by [64]. The best bound for radio gossip known so far, however, is $O(D + \Delta \log n)$ steps in an n-node graph with diameter D and maximum degree Δ [37], even though there is no relation in general between the optimum radio gossip time and the maximum degree. Indeed, for general graphs, there is a polynomial inapproximability lower-bound for the minimum time radio gossip problem [47].

Planar path separators. For our results on planar graphs, we rely on the structure of path-separators. Lipton and Tarjan first found small $O(\sqrt{n})$-sized separators for n-node undirected planar graphs [62]. More recently, planar separators based on any spanning tree of a planar graph were found [79] with the following key property: these balanced vertex separators can be formed by starting at any vertex and taking the union of three shortest paths from this vertex. Minor-free graphs also admit small path-separators as found by [1]; in this case, the number of paths used depends on the graphs which are excluded minors, but stays constant for constant-sized excluded minors.

3.2 LP Rounding for Multicast in General Graphs

In this section we present an approximation algorithm for finding a **minimum poise Steiner subgraph**, and establish an LP integrality gap upper bound, thus proving Theorem 3.1.4. We begin by presenting a linear program for a multicommodity generalization of minimum poise Steiner subgraph, which is useful for the multicommodity multicast problem. This linear program, when specialized to the case where we need to connect a root r to a subset R of terminals, is our LP for the minimum poise Steiner subgraph problem.

3.2.1 Linear Program for Poise

The generalized Steiner poise problem is to determine the existence of a subgraph containing paths for every demand pair in $K = \{(s_i, t_i) | 1 \leq i \leq k\}$ of poise at most L, i.e. every demand pair is connected by a path of length at most L and every node in the subgraph has degree at most L.

We use indicator variables $x(e)$ to denote the inclusion of edge e in the subgraph. Since the poise is at most L, this is also an upper bound on the length of the path from any terminal to the root. For every terminal $(s_i, t_i) \in K$, define $\mathcal{P}_i$ to be the set of all (simple) paths from s_i to t_i. We use a variable $y_t(P)$ for each path $P \in \mathcal{P}_i$ that indicates whether this is the path used by s_i to reach t_i in the subgraph. For a path P, let $\ell(P)$ denote the number of hops in P. The integer linear program for finding a subgraph of minimum poise is given below.

$$
\begin{array}{lll}
\text{minimize} & L = L_1 + L_2 & (POISE - LP) \\
\text{subject to} & \sum_{e \in \delta(v)} x(e) \leq L_1 & \forall v \in V \\
& \sum_{P \in \mathcal{P}(t,r)} y_t(P) = 1 & \forall t \in R \\
& \sum_{P \in \mathcal{P}(t,r)} \ell(P) y_t(P) \leq L_2 & \forall t \in R \\
& \sum_{P \in \mathcal{P}(t,r): e \in P} y_t(P) \leq x(e) & \forall e \in E, t \in R \\
& x(e) \in \{0, 1\} & \forall e \in E \\
& y_t(P) \in \{0, 1\} & \forall t \in R, P \in \mathcal{P}_L(t, r).
\end{array}
$$

The first set of constraints specifies that the maximum degree of any node using the edges in the subgraph is at most L_1. The second set insists that there is exactly one path chosen between every pair $(s_i, t_i) \in K$. The third set ensures that the length of the path thus selected is at most L_2. The fourth set requires that if the path $P \in \mathcal{P}_i$ is chosen to connect s_i to t_i, all the edges in the path must be included in the subgraph.

We will solve the LP obtained by relaxing the integrality constraints to nonnegativity constraints[1], and get an optimal solution $x, y \geq 0$.

For the remainder of this section, we will focus on the rooted version of this problem. In particular, there will be a root r and set of terminals R, then we will make $K = \{(r, t) | t \in R\}$. It still remains to round a solution to POISE-LP to prove Theorem 3.1.4. Before presenting the rounding algorithm in Section 3.2.3, we describe a result on multiflows that will be useful in decomposing our LP solution into a set of paths that match terminals with each other.

3.2.2 Preliminaries

Given an undirected multigraph G with terminal set $T \subset V$ of nodes, a *multiflow* is an edge-disjoint collection of paths each of which start and end in two distinct terminals in T. The value of the multiflow is the number of paths in the collection. Such a path between two distinct terminals is called a T-path and a multiflow is called a T-path packing. For any terminal $t \in T$, let $\lambda(t, T \setminus t)$ denote the minimum cardinality of an edge cut separating t from $T \setminus t$ in G. Note that in any multiflow, the maximum number of paths with t as an endpoint is at most $\lambda(t, T \setminus t)$. Furthermore,

[1]Even though the number of path variables is exponential, it is not hard to convert this to a compact formulation on the edge variables that can be solved in polynomial time. See e.g., [70]

since every path in a multiflow has to end in distinct vertices in T, the maximum value of any multiflow for T is upper bounded by $\sum_{t \in T} \frac{\lambda(t, T \setminus t)}{2}$, by summing over the maximum number of possible paths from each terminal and dividing by two to compensate for counting each path from both sides. This upper bound can be achieved if a simple condition is met.

Theorem 3.2.1. [15, 63] If every vertex in $V \setminus T$ has even degree, then there exists a multiflow for T of value $\sum_{t \in T} \frac{\lambda(t, T \setminus t)}{2}$.

The following simple construction will be useful in the rounding algorithm to identify good paths to merge clusters. It is based off of a lemma from [70].

Lemma 3.2.2. Let G be a digraph where every node has at most one outgoing edge (and no self loops). In polynomial time, one can find an edge-induced subgraph H of G such that H is a partition of the nodes of G into a forest of directed trees each being an inward arborescence, and with $|E(H)| \geq |E(G)|/2$.

Proof. Consider any connected component of G, if there are v vertices, then there are either v or $v-1$ edges (as each vertex has outdegree at most 1). If there are v edges, there is a cycle. When we remove an edge from the cycle, we now have a connected component with $v-1$ edges. If there are $v-1$ edges and all the vertices have outdegree at most 1, then it is already an inward arborescence.

An algorithm to find H would simply check the graph for directed cycles, and if any cycle exists, it would remove an edge from that cycle. Any component which had a cycle has at least 2 edges, and we remove at most 1 edge from every component. So, the resulting graph H has at least half the edges that the original graph G had. □

3.2.3 The Rounding Algorithm

The main idea of Algorithm 1 is to work in $O(\log k)$ phases, reducing the number of terminal-containing components in the subgraph being built by a constant fraction at each stage [72]. We begin with an empty tree containing only the terminals R, each in a cluster by themselves. In each phase, we will merge a constant fraction of the clusters together carefully so that the diameter of any cluster increases by at most an additive $O(L)$ per phase: for this, we choose a terminal as a center of each cluster. When we merge clusters, we partition the clusters into stars where we have paths of length $O(L)$ from the centers of the star leaf clusters to the center of the star center-cluster. These steps closely follow those in [70]. The crux of the new analysis is to extract a set of stars that merge a constant fraction of the current cluster centers using a solution to POISE-L LP.

The key subroutine to determine paths to merge centers is presented in Algorithm 2. This uses the multiflow packing theorem of [15, 63].

3.2.4 Performance Ratio

In this section, we prove Theorem 3.1.4. The performance ratio of the rounding algorithm in the theorem is a consequence of the following claims, the first of which follows directly from the path pruning in Algorithm 2.

Lemma 3.2.3. The length of each path output by Merge-Centers($\mathcal{C}^*$) is at most $4L$.

Algorithm 1 LP Rounding for Poise-L tree

1: Clusters $\mathcal{C} \leftarrow R$; Centers $\mathcal{C}^* \leftarrow R$; Solution graph $H \leftarrow \emptyset$; Iteration $i \leftarrow 1$.
2: **while** $|\mathcal{C}| > 1$ **do**
3: Use Algorithm Merge-Centers($\mathcal{C}^*$) to identify a subgraph F_i whose addition reduce the number of clusters by a constant fraction;
4: $H \leftarrow H \cup F_i$; Update $\mathcal{C}$ to be the set of clusters after adding the subgraph F_i, and update $\mathcal{C}^*$ to be the centers of the updated clusters based on the star structure from Algorithm Merge-Centers($\mathcal{C}^*$). Increment i.
5: **end while**
6: Add a path of length at most L from r to the center of the final cluster in H. Find a shortest path tree in H rooted at r reaching all the terminals in R and output it.

Lemma 3.2.4. The expected number of paths output by Merge-Centers($\mathcal{C}^*$) is $\Omega(|\mathcal{C}^*|)$.

Proof. The main observation here is that the total number of edges in the multigraph G is at most $|\mathcal{C}^*| \cdot L \cdot M$. To see this, note that each terminal t retains its flow of value 1 in POISE-LP corresponding to the paths with nonzero value for y_t. Thus in the scaled version, it retains M paths to the root, and the average number of hops in these paths is at most $L_2 \leq L$ hops, for each terminal. Summing over all terminals, the number of edges in G is at most $|\mathcal{C}^*| \cdot L \cdot M$.

The total number of paths discarded cannot exceed $\frac{|\mathcal{C}^*| \cdot M}{4}$. Otherwise the paths each of length at least $4L$ each would need more edges in G than we started with. After discarding, we expect to still collect at least $\frac{1}{M} \cdot (|\mathcal{C}^*| \cdot M - \frac{|\mathcal{C}^*| \cdot M}{4}) = \frac{3|\mathcal{C}^*|}{4}$ paths fractionally. Hence the expected number of terminals in the subgraph H is at least $\frac{3|\mathcal{C}^*|}{4}$. The set of arcs finally retained in H'' is at least one third of the nodes of H, the worst case being a path of two arcs. This leads to an expectation of at least $\frac{|\mathcal{C}^*|}{4}$ paths finally output to merge the initial clusters. □

Lemma 3.2.5. The distance of any node in a cluster to its center increases by at most $4L$ in the newly formed cluster by merging paths corresponding to stars in H''. Thus, the diameter of any cluster in iteration i is at most $8iL$.

Proof. The proof is by induction over i, and is immediate by observing that any node can reach the new merged cluster center say c by first following the path to its old center, say t and then following the path P_t corresponding to the arc in H'' from t to c. By Lemma 3.2.3 above, the length of P_t is at most $4L$ and the claim follows. □

Lemma 3.2.6. The maximum degree at any node of G induced by the union of paths output by Merge-Centers($\mathcal{C}^*$) is $O(L)$.

Proof. This is a simple consequence of the performance guarantee of rounding the LP solution obtained for the collection of paths. Since the paths we found pack into the LP solution $2x$ (from the property of the multiflow packing), the expected congestion due to the chosen random paths on any edge e is at most $2x(e)$. From the first constraint in the LP, the expected congestion at any node due to paths incident on it is at most $2L_1 \leq 2L$, by linearity of expectation.

Algorithm 2 Merge-Centers($\mathcal{C}^*$) using LP solution x

1: Multiply the POISE-LP solution x by the least common multiple M of the denominators in the nonzero values of x to get a multigraph.
2: For every terminal $t \in \mathcal{C}^*$, retain the edges in the paths corresponding to the paths in its LP-solution with nonzero value (i.e., paths P with nonzero $y_t(P)$), for a total of M connectivity from t to r. Note that the union of all the retained edges gives connectivity M from every $t \in \mathcal{C}^*$ to r and hence by transitivity, between each other.
3: Double each edge in the multigraph and use Theorem 3.2.1 to find a multiflow of value $\sum_{t\in\mathcal{C}^*} \frac{\lambda(t,\mathcal{C}^*\setminus t)}{2} \geq \sum_{t\in\mathcal{C}^*} \frac{2M}{2} = |\mathcal{C}^*| \cdot M$. Note that each terminal in $\mathcal{C}^*$ has at least M paths in the multiflow.
4: Eliminate all the paths in the multiflow of length longer than $4L$.
5: For every terminal t, pick one of the M paths incident on it uniformly at random and set this path to be P_t. If the chosen path is eliminated due to the length restriction, set $P_t \leftarrow \emptyset$.
6: Let H be an auxiliary graph on vertex set $\mathcal{C}^*$ with at most one arc coming out of each $t \in \mathcal{C}^*$ pointing to the other endpoint of P_t (or add no edge if $P_t = \emptyset$).
7: Apply Lemma 3.2.2 to the subgraph of H made of nodes, to get a collection H' of in-trees. For each in-tree, partition the arcs into those in odd and even levels of the tree and pick the set with the larger number of arcs. Note that these sets form stars originating from a set of centers and going to a single center. Let H'' denote the set of these stars.
8: For each arc of the stars in H'', include the path P_t originating at the leaf of the star corresponding to the arc in H'', and output the collection of paths.

We apply the classic rounding algorithm of [52]. Since the length of each path in the collection is at most $4L$ and the expected congestion is at most $2L$, we obtain that there is a rounding, which can be determined in polynomial time, such that the node congestion (degree) in the rounded solution of at most $4L$. $4L$. □

By Lemma 3.2.4, the number of iterations of the main Algorithm 1 is $O(\log k)$ where k is the number of terminals. Lemma 3.2.5 guarantees that the subgraph of the final cluster containing all the terminals has distance $O(\log k \cdot L)$ between any pair of terminals. Since the final output is a shortest path tree of this subgraph rooted at r, its diameter is also of the same order. Lemma 3.2.6 ensures that the total degree of any node in the subgraph of the final cluster is $O(\log k \cdot L)$, and this is also true for the tree finally output. This completes the proof of Theorem 3.1.4. We can derandomize the above randomized algorithm using the standard method of pessimistic estimators [65].

3.3 Approximating multicommodity multicast on planar graphs

In this section we prove Theorem 3.1.1. Let $G = (V, E)$ be the given planar graph, with $n = |V|$, and let $K = \{(s_i, t_i) : 1 \leq i \leq k\}$ be the set of the k source-destination pairs that need to be connected. Let $\gamma = 1/\log k$. We given a brief overview of our algorithm PlanarMCMulticast, which is fully described in Algorithm 3.

PlanarMCMulticast is a recursive algorithm, breaking the original problem into smaller problems each with at most a constant fraction of the demand pairs in K in each recursive call, thus having $O(\log k)$ depth in the recursion. For a given graph, the algorithm proceeds as follows.

- Find a node separator composed of three shortest paths from an arbitrary vertex [79] to break the graph into pieces each with a constant fraction of the original nodes.
- Solve a generalized Steiner poise LP on the given pairs to identify demand pairs that cross the separator nodes to an extent at least $\Omega(\gamma)$.
- Satisfy these demand pairs by routing their messages from the sources to the separator, moving the messages along the separator (since they are shortest paths, so this movement takes minimal time) and back to the destinations, by scaling the LP values by a factor of $O(\frac{1}{\gamma})$ and using Theorem 3.1.4 to find a low poise tree to route to/from the separator.
- For the remaining demand pairs (which are mainly routed within the components after removing the separators), PlanarMCMulticast recurses on the pieces.

The key aspect of planarity that is used here is the structure theorem that planar graphs contain [79] small-size balanced vertex separators that are a combination of three shortest paths starting from any given node.

We now prove that PlanarMCMulticast constructs, in polynomial-time, a multicommodity multicast schedule a schedule of length $O((OPT \log^3 k \cdot \frac{\log n}{\log \log n})$ where OPT is the length of the optimal schedule.

1. Observe that $3OPT$ is an upper bound for the value L for the POISE-LP for this instance.
2. In Step 2 of PlanarMCMulticast, the separator is obtained using the algorithm in [79]. In Step 3, we use POISE-LP, as specified in Section 3.2 to find the fractional solution. In Step 4 of PlanarMCMulticast, unit $s_i - t_i$ flow is achieved by scaling up the LP cost by at most a factor of $3/\gamma$ since at least $\gamma/3$ flow intersects one of the three paths in $\mathcal{P}$. Now, observe that this scaled LP solution immediately yields a valid solution to POISE-LP in Step 5. Applying Theorem 3.1.4, in Step 6, with the value of $L = O(OPT \log k)$ gives a tree of poise $O(OPT \log^2 k)$.
3. The algorithm performs $O(\log k)$ (recursive) phases; the poise of the tree at the ith level of recursion is itself based on an LP that has been scaled by a factor of at most $\frac{1}{1-\gamma}$ (in Step 8) in the previous $i - 1 = O(\log k)$ iterations followed by a final scaling of $\frac{3}{\gamma}$ in the last iteration. In any iteration, the total factor by which the initial LP value of OPT is scaled is at most $(\frac{1}{1-\gamma})^{i-1} \cdot \frac{3}{\gamma} \leq (1+\gamma)^{\frac{1}{\gamma}} \cdot \frac{3}{\gamma} = O(\log k)$ since $\gamma = \frac{1}{\log k}$.
4. In Step 7, we incur a multiplicative factor of $O(\frac{\log n}{\log \log n})$ in going from a small poise tree to a schedule. Here, we crucially use the fact that the separator paths are shortest paths - for a demand pair (s_i, t_i) let f_i denote the first vertex on the separator path that the message arrives at after leaving source s_i and let l_i denote the last vertex (on the separator path) that the message departs from, on its way to the destination t_i; then f_i and l_i must be at most an additive $O(OPT)$ of the sum of the lengths of the paths from s_i to f_i and l_i to t_i along the separator path, since every demand pair has a path of length $O(OPT)$ between them in the LP solution in this subgraph. Thus in Step 7, we can wait to aggregate all messages from the

Algorithm 3 PlanarMCMulticast(G, K)

1: **Base case:** When $K = \{(s_1, t_1)\}$ has one demand pair, schedule the message on the shortest path between the source, s_1, and destination, t_1.
2: **Separate the graph:** Define the weight of a node as the number of source-destination pairs it is part of, and the weight of a subset of nodes as the sum of their weights. Find a 3-path separator $\mathcal{P}$ of G, given by shortest paths P_1, P_2, and P_3, whose removal partitions the graph into connected components each of which has weight at most half that of the graph [79].
3: **Partition the terminal pairs:** Partition the set K into two subsets, by solving the POISE-LP.
- Let K_1 consist of pairs (s_i, t_i) such that in POISE-LP, the fraction of the unit flow from s_i to t_i that intersects $\mathcal{P}$ is at least γ.
- Let $K_2 = K - K_1$ consist of the remaining pairs, i.e. pairs (s_i, t_i) such that in the LP, the fraction of the unit flow from s_i to t_i that intersects $\mathcal{P}$ is less than γ

4: **Scale flow for pairs in** K_1: For each pair (s_i, t_i) in K_1, scale the flow between s_i and t_i in the POISE-LP by $\frac{3}{\gamma}$ so there exists a path P_j which intersects a unit of this scaled s_i-t_i flow; remove other s_i-t_i flows that does not intersect P_j up to a unit. Assign (s_i, t_i) to a set S_j.
5: **Create 3 minimum poise Steiner tree problems for** K_1: For each path P_j, create a minimum Steiner poise problem as follows: (i) attach, to the graph, an auxiliary binary tree T_j with nodes of P_j forming the leaves, and adding new dummy internal nodes (This step is similar to [66]); (ii) set the root of the binary tree to be the root for the Steiner poise problem, and the terminals to be all the s_i and t_i in S_j.
6: **Round the POISE-LP solution**: For each P_j, round the LP to obtain a Steiner tree T_j of small poise connecting all the terminals in S_j with the root using the algorithm from Theorem 3.1.4.
7: **Construct schedule for** K_1: Use Lemma 3.1.3 on the tree T_j to perform a multicast between all terminals in it as follows: use the multicast schedule to move the messages, from the sources, till they hit the path P_j, then move messages along the path followed by the multicast schedule in reverse to move them towards the destinations. (Moving messages along a path can be achieved by a schedule that alternates between the even and odd matchings in the path for as many steps as the target length of the schedule)
8: **Scale flow for** K_2: For each pair (s_i, t_i) in K_2, remove any flow that intersects $\mathcal{P}$ and scale the remaining flow (by a factor of at most $\frac{1}{1-\gamma}$) so as to continue to have unit total flow between the pair.
9: **Recurse for** K_2: For each connected component C_j, let K_2^j denote the subset of K_2 with both terminals in C_j. Run PlanarMCMulticast(C_j, K_2^j) in parallel.

sources at the separator path, then move all the messages one way along the path and then the opposite way, for as many time steps as the poise of the integral tree, without more than tripling the total schedule.

5. Since there are $O(\log k)$ recursive phases, the final schedule has length $O(OPT \log^3 k \frac{\log n}{\log \log n})$.

This proves Theorem 3.1.1.

3.4 A polylogarithmic approximation for radio gossip on planar graphs

In this section, we present an $O(\log^2 n)$-approximation algorithm for finding a radio gossip schedule on planar graphs, and prove Theorem 3.1.5.

Let $G = (V, E)$ be a given planar graph. Once the messages from all nodes have all been gathered together at a node we can easily broadcast them back out in $O(OPT + \log^2 n)$ rounds using [59]. So we focus on gathering the messages together at one node. To do this, we recursively find 3-path separators in the graph [79] to decompose it into connected components. Then, working backwards, we gather messages from the 3-path separators found in an iteration at the nodes of the 3-path separators found in previous iterations, using techniques from telephone multicast [66]. The key properties used in the recursive algorithm are that the number of paths in the separator is a constant 3 and the paths are all shortest paths in the component they separate from some vertex.

We assume the optimal schedule has length L. Note that $L \leq 2n$ since gossip can be achieved by simply choosing any spanning tree and broadcasting one node at a time in post-order (to gather all messages at the root) and then in reverse post-order (to spread all messages back to all nodes). We also assume that $L > 2$ ($L \leq 2$ only occurs if there are 1 or 2 nodes total). The details of the algorithm are given in Algorithm 4.

We will first prove a couple lemmas needed for the proof of Theorem 3.1.5.

Lemma 3.4.1. The graph G'_i has a $3L$ matching which matches ever vertex of N_i to a some vertex of U_i and every vertex of U_i has degree at most $3L$.

Proof. Consider the graph $G'_i = (N_i, U_i, E'_i)$. Let p_v be a path that v's message take from v to r in the optimal solution. Let p'_v be the prefix of p_v until the first vertex of U_i. All the paths p_v have length at most L (since this is the length of the optimal schedule). For each node $w \in V_i$, w is in at most one of the p'_v for $v \in N_i$. This is because if two p'_v's from the same path in P_i arrive at a node, there would be a path of length at most $2L$ between two nodes in N_i from the same path P_i; But the paths in P_i were chosen to be shortest paths in $\mathcal{C}_{i-1}$, and N_i were nodes that were pairwise distance $2L+1$ from each other, a contradiction. Now consider $u \in U_i$ and the p'_v for N_i that match to u: there can be at most L nodes from which messages go from V_i to u (since that is an upper bound on its message receiving degree in the optimal solution). Thus, in the optimal solution there are at most $3L$ paths from N_i to any specific node in U_i and these paths have length at most L. So, there must exist a $3L$-matching in G' which matches every vertex of N_i to some vertex of U_i and no vertex of U_i has degree more than $3L$.

□

Lemma 3.4.2. Each iteration of the step 13 in algorithm 4 takes time at most $O(L \log n)$ and moves the messages to U_i.

Proof. In Algorithm 4, step 13 is just to achieve the last step of message movement in the q_v paths. Each node $w \in V_i$ can be adjacent to at most 3 nodes in a given path of P_j for any $j < i$, as these paths are shortest paths in G_{j-1}, and in particular these nodes are within 2 of each other in the path. Also, for a given $w \in V_i$, w can be adjacent to multiple paths in P_j but they must all be in the same

Algorithm 4 A gathering procedure for radio gossip in planar graphs.

Clusters $\mathcal{C}_0 \leftarrow \{V\}$; Vertices $V_0 \leftarrow V$; Graph $G_0 \leftarrow G$; Iteration $i \leftarrow 1$.
while $V_{i-1} \neq \emptyset$ **do**
  for all connected component $C \in \mathcal{C}_{i-1}$ **do**
    Choose some $v \in C$. Find shortest paths p_1, p_2, p_3 from v that form a 3-path separator in C using [79]; Add these to P_i, the paths found in the ith iteration.
    Add v and every $(2L+1)$st vertex along paths p_1, p_2, p_3 to N_i
  end for
  Remove the vertices in P_i from V_{i-1} to get V_i; Let G_i be $G[V_i]$ and $\mathcal{C}_i$ denote the connected components of G_i; Increment i.
end while
while $i > 0$ **do**
  Do $2L$ rounds of radio broadcasts in series on nodes that are $2L+1$ apart from each other along the paths in P_i to gather all the messages on P_i at the nodes N_i.
  Form G'_i a bipartite graph from N_i to $U_i = \cup_{j=1}^{i-1} P_j$. Add an edge $uv \in E'_i$ if there is a path from $u \in N_i$ to v in $G[\mathcal{C}_{i-1} \cup \{v\}]$ of length at most L. Find a $3L$-matching in G' where every vertex of U_i has degree at most $3L$.
  Do up to L rounds of radio broadcast to get the messages from N_i to within one node of U_i, along the paths in the $3L$-matching found above. Note that the messages stay within the component in $\mathcal{C}_{i-1}$ containing u for this part.
  Move the messages from the last nodes in $\mathcal{C}_{i-1}$ to their destination nodes in U_i in the $3L$-matching using at most $9L$ rounds for each of the paths ($27L \log n$ total).
  Decrement i
end while

component of $\mathcal{C}_{j-1}$, and there is at most three such paths. Let $S_j^k(\ell)$ be every third vertex on the paths P_j^k starting with the ℓth vertex. In $3L$ steps (the maximum degree of the matching at these nodes) we can gather the messages that need to be received at $S_j^k(\ell)$ as no node is adjacent to two nodes in this set. Doing this gathering for every shift ℓ from one to three, and each of three choice of which path k, a total of $27L$ steps gets the message from the q_v to P_j. This process is repeated for each collection P_j with $j < i$. Now all the messages that were along P_i have been moved to some node in $U_i = \cup_{j=1}^{i-1} P_j$ in $27L \log n$ steps.

□

Having established the lemmas, we now give the proof of Theorem 3.1.5.

Proof of Theorem 3.1.5. First, we establish that the algorithm runs correctly. Let r be the root (chosen in the first iteration). First the algorithm gathers the messages on the P_i to N_i. We will divide the paths into P_i^1, P_i^2, P_i^3, so that each component of $\mathcal{C}_{i-1}$ that has three paths puts one path in each of these sets. Now, we will handle each of the P_i^j one at a time. To deal with P_i^j, in the kth step, the nodes which are $2L-k$ further from a node in N_i broadcast all their messages to the node one closer to N_i. There is no interference amongst the nodes in V_{i-1} as only nodes at distance at least $2L$ in each component of $\mathcal{C}_{i-1}$ are broadcasting at a time, and nodes in different components

are non-adjacent (since they are disconnected by the separators P_{i-1}). This will gather all the messages along P_i^j. Doing this once for each of the P_i^j in $2L$ steps, all the messages currently on P_i will be gathered at the N_i in $6L$ steps. The messages are all currently at N_i or U_i.

Lemma 3.4.1 tells us that G_i' has a $3L$-matching as desired, and so we can find such a matching. Once a matching is chosen, let q_v be a shortest path within G_{i-1} from $v \in N_i$ to the vertex it is matched with in U_i. Within each component of $\mathcal{C}_i$, we can broadcast along the q_v for every N_i in one of the paths simultaneously; There will be no interference as the N_i's and their matching paths are far apart within $\mathcal{C}_{i-1}$. Thus, it takes at most L rounds of radio broadcasting to move the messages from N_i along their q_v to the vertex before U_i.

Lemma 3.4.2 gives us that in time $O(L \log n)$ we move the messages onto U_i.

In the last iteration of the process, we will have all the messages on the first path separator P_1. P_1 is a path separator of shortest paths on the whole graph and the diameter of G is a lower bound on L. So, in $3L$ steps we can move the messages from P_1 to r. We have now successfully gathered all the messages to r.

The time it takes to deliver all the messages to r is at most $O(L \log^2 n)$. The path separator ensures that each component has at most a constant fraction of the number of vertices of the original graph. Therefore, the final $i \leq \log n$. Each iteration, the number of rounds of broadcast we do is $6L$ in the first part and $27L \log n$ in the last step. So, this schedule uses $O(L \log^2 n)$ steps to gather all the messages at r. □

3.5 Minor-free graphs

In this section, we prove that both results on planar graphs can be extended to any family of minor-free graphs. For this section, we will have to use the more general definition for path-separators.

Definition 3.5.1. [1] A vertex-weighted graph G is k-path separable if there exists a subgraph S, called a k-path separator, such that:

1. $S = P_0 \cup P_1 \cup \ldots$ where each subgraph P_i is the union of k_i shortest paths in $G \backslash \cup_{j<i} P_j$
2. $\sum_i k_i \leq k$
3. either $G \backslash S$ is empty, or each connected component of $G \backslash S$ is k-path separable and has total vertex weight at most half of the original.

This definition of path separator while more complicated can be integrated into our PlanarMCMulticast algorithm and algorithm 4 with only small adjustments. We will use the following to theorem which tells us when a k-path separator exists and can be found.

Theorem 3.5.2. [1] Every H-minor-free weighed connected graph is $k(H)$-path separable, and a $k(H)$-path separator can be computed in polynomial time.

Note that $k(H) = O(|H|)$ in the above construction.

3.5.1 Multicommodity multicast in minor-free graphs

Consider that our graph is H-minor free and $k(H)$ is the number of paths needed for the path separator. We will need to repeat steps 3-8 of the original algorithm for each subgraph of shortest

paths. Other than that, the algorithm remains the same.

The main changes to the algorithm are as follows.

- The path separator we find is different.
- We may need to iterate through steps 3-8 of the original algorithm multiple times.

Since these are the only changes to the algorithm, we first see that finding the path separator takes polynomial time, and the number of iterations increases by at most a factor of $k(H)$ (a constant) so the algorithm still runs in polynomial time.

The only other change to the analysis is that the number of recursions is $k(H)\log k$. This gives that we incur a factor of $(\frac{1}{1-\gamma})^{k(H)\gamma}\frac{3}{\gamma}$ when scaling the LP. Since $\gamma = \frac{1}{\log k}$, the LP gets scaled by a factor of $O(e^{k(H)}\log k)$. Since $|H|$ and hence $k(H)$ is a constant, we get that the resulting schedule from this algorithm is at most $O((OPT\log k + \log n)\log^2 k\frac{\log n}{\log\log n})$. In terms of $k(H)$, our algorithm builds a schedule that takes $O((OPT\log k + \log n)\log^2 k\frac{\log n}{\log\log n}k(H)e^{k(H)})$

This proves theorem 3.1.6.

3.5.2 Radio Gossip in Minor-free Graphs

The modification to go from radio gossip on planar graphs to radio gossip on minor-free graphs is even more simple. Let $k = k(H)$ where our graph G is H-minor-free. We only change how we set up all of our initial sets; P_i, C_i, V_i, N_i, U_i. For each iteration, we will define up to k of these sets one for each of the (potential) subgraphs which compose the path separator.

The only major change to this algorithm is the set-up. In the set-up, we have to process the subgraphs which compose the path separator one at a time (as opposed to there only being one subgraph which is the whole path separator). This only increases the number of iterations by a factor a $k(H)$.

The other change is that everywhere we had a 3 before it now becomes a $k = k(H)$. All our previous lemmas and theorems hold if we change the 3 arising from the planar case to k. Therefore, we have shown that this algorithm produces a schedule for gathering which runs in time $O(L\log^2 n)$ (or $O(Lk^3\log^2 n)$ if k is not constant) to gather all the messages in one place.

We again use the result of Kowalski and Pelc to broadcast the messages once they have been gathered [59]. This broadcast takes time $L + O(\log^2 n)$, so the whole gossip schedule takes time $O(L\log^2)$ proving Theorem 3.1.7 as desired. We again use the result of Kowalski and Pelc to broadcast the messages once they have been gathered [59]. This broadcast takes time $L + O(\log^2 n)$, so the whole gossip schedule takes time $O(L\log^2 n)$ proving Theorem 3.1.7 as desired. If we don't assume $k = k(H)$ is a constant, then the algorithm takes time $O(Lk^3\log^2 n)$.

3.6 Conclusion

We gave the first proof of an upper bound of $O(\log k)$ on the integrality gap of the POISE-LP. We utilized the poise result combined with path separators to develop an $O(\log^3 k \frac{\log n}{\log \log n})$-approximation for telephone multicommodity problem on planar graphs. In addition, we develop an $O(\log^3 n)$-approximation for radio gossip on planar graphs. Lastly, we extend these results to graphs which are minor-free. One natural open problem is if these path separator techniques can extend to the other communication problems in planar graphs. In particular, does the path separator technique or poise problem allow for better approximations for the edge-star model in planar graphs.

Chapter 4

Design of Overlapping Networks

4.1 Introduction

In large Internet publishing systems, a variety of sources of information constantly refresh their content, while a set of subscribers continuously pull this updated information. The recent widespread adoption of the "Internet of Things" and "Web 3.0" tools similarly involves the constant real-time sharing of information between producers of relevant content and their corresponding consumers. Common examples include syndication systems as well as distributed databases that contain information originating at sources with sinks interested in the most up to date copies.

A natural approach to enable efficient information transfer in such systems is to build a *cost-effective* collection of networks, one for each publisher and supplier: the publishers push their updates to a set of locations via their respective networks, while the subscribers pull the information, refreshed by multiple publishers, from these intermediate nodes using their own networks. Note that each subscriber network needs only to *overlap* those publishers' networks that are of interest. Such interests are represented by an auxiliary bipartite *demand* graph with publishers on one side, subscribers on the other, and edges (of interest) between the two. Since the individual networks are being used for scatter/gather or push/pull operations (by publishers/subscribers respectively) the two natural structures are: trees and overlay stars. Trees correspond to situations where the entity (e.g. a pusher, such as Facebook, or a puller, such as the IRS) has sufficient network presence to employ multicast/reverse-multicast while overlay stars correspond to point-to-point communication.

This basic framework gives rise to a class of problems we have christened *DON* or Design of Overlapping Networks. Given their relevance to developments in the Internet ecosystem, these theoretical problems are significant from a practical perspective. Our central goal is to settle the polynomial-time approximability for the most general DON problem, in which we have an arbitrary demand graph, and arbitrary choice of tree or overlay star by each publisher/subscriber. In this chapter, we obtain a constant approximation for the special case when all subscribers are interested in all publishers, and a logarithmic approximation for the general case. The latter approximation is, in fact, with respect to the value of a natural linear programming relaxation of the problem. In a contrasting result, we establish a non-constant integrality gap for this linear program. However,

the exact approximability status of the general DON problem remains tantalizingly open.

4.1.1 Problem Definition

In the general DON problem we are given an undirected graph $G = (V(G), E(G))$ with non-negative costs on the edges $c : E \to \mathbb{Z}^+$, subsets of nodes $P, S \subseteq V$ (publishers and subscribers respectively), the type of network to be installed for each publisher and subscriber, $Type : P \cup S \to \{\text{tree}, \text{star}\}$, and an auxiliary demand graph $D = (V(D), E(D)$ where $V(D) = P \cup S$ and $E(D) \subseteq P \times S$ specifying (publisher, subscriber) pairs whose networks are required to overlap (intersect); the goal is to build a collection of networks satisfying the input requirements. We assume that the edge costs form a metric: they are symmetric and satisfy the triangle inequality. Any instance of the general DON problem can be split into four sub-instances. When the type requirement $Type$ is tree (resp., star) for all publishers and subscribers we refer to the problem as *tree-tree DON* (resp. *star-star DON*). We also use *tree-star DON* to refer to the two problem variants where on one side, say the publishers, we have rooted trees while on the other, we have rooted stars. We use the prefixes *general* and *complete* to denote arbitrary and complete demand graphs, respectively, as in general tree-tree DON or complete tree-star DON, etc. Thus, the term general (complete) DON refers to the problem where the demand graph is arbitrary (complete) and the type requirement may vary across terminals.

We denote the installed network by $N = (V, E_N)$; P_i denotes the network installed at publisher p_i and S_j the network rooted at subscriber s_j. Then, the multi-graph $N = (\cup_{p_i \in P} P_i) \cup (\cup_{s_j \in S} S_j)$ is the (multi-set) union of all the installed networks. The cost of N is the sum of the costs of all the constituent networks, with each edge counted as many times as the number of individual networks they are present in. Recall that the installed networks are operated autonomously by each publisher and subscriber, and thus the cost of an edge needs to be multiplied by the number of such independent networks that build and utilize it in their updates.

4.1.2 Results and Techniques

We present new algorithms and results for several DON problems.

1. We conjecture that a polynomial-time constant-factor approximation for general DON is not achievable. We present in Section 4.2.1 an $\Omega(\log \log n)$ integrality gap for a natural LP relaxation of the general tree-tree DON; note that this result also extends to the general DON problem. This integrality gap proof, which is our strongest technical contribution, is based on a novel reduction from a well-studied LP relaxation for the group Steiner problem, applied to a hypercube demand graph instance of DON.

   On the positive side, we present an $O(\log n)$-approximation algorithm for the general DON problem in Section 4.2.2. The main ingredient of our result is a constant-factor approximation algorithm for tree-star DON on tree metrics, by a careful deterministic rounding of an LP relaxation of the problem. The logarithmic approximation for general DON follows by extending to general metrics and combining with results for the star-star and tree-tree variants.

2. We next study the complete DON variants where the demand graph is complete. We give constant-factor approximation algorithms for all three variants— tree-tree, star-star and tree-star— which together yield a constant-factor approximation for complete DON. Unlike our algorithm for general DON, all of our algorithms for the complete demand case are combinatorial; they combine structural characterizations of near-optimal solutions with interesting connections to access network design and facility location problems.

   (a) Our approximation factor for complete tree-tree DON in Section 4.3.1 is $4\rho_{TS}$ where ρ_{TS} is the best known approximation for the tree-star Access Network Design problem (which is generalized by the Connected Facility Location or CFL problem). Thus $\rho_{TS} \leq \rho_{CFL} \leq 4$ [19].

   (b) Our approximation factor for complete star-star DON in Section 4.3.2 is 4α, where α is the best approximation factor achieved for uncapacitated facility location, improves over the result in [10].

   (c) For the complete tree-star-DON problem, we get a $4\rho_{TS}$-approximation in Section 4.3.3.

3. Lastly, we consider when the publishers and subscribers are the same. In this case, there is a flow proof that the optimal solution has one node to which all the other nodes take a shortest path to. In Section 4.5, we provide a proof of this case, and a different approximation algorithm for the complete tree DON case which has an improved approximation factor when P and S are close to the same size.

4.1.3 Related Work

Data Dissemination Networks. Our formulation of DON generalizes network data dissemination problems first studied in [10]. Using our terminology, the relevant results of [10] are $O(\log n)$-approximation algorithms for general tree-tree DON and general star-star DON, and a 14.57-approximation for the complete star-star DON. Our work improves the approximation factor for complete star-star DON to under 6 (since the current best approximation for uncapacitated facility location is 1.488 [61]), and presents new results for many other DON problems. The star-star DON problem is also closely related to the minimum-cost 2-spanner problem studied in [17, 58]. In particular, a greedy algorithm essentially along the same lines as an algorithm of [17] yields an $O(\log n)$-approximation for the star-star-DON problem even when the underlying distances do not form a metric.

Network Design. There has been considerable work in network design, which is concerned with the design of network structures that satisfy some connectivity properties and optimize some underlying cost structure [82]. Well-known problems in this area include the minimum Steiner tree [34], group Steiner tree [35], and general survivable networks [81]. One key distinction between many of these network design problems and DON is that the desired solution in DON is a collection of networks (as opposed to a single network), and each edge contributes to the total cost of the solution as many times as it occurs in the network collection. On the other hand, the goal in many classical network design problems is to build a single network. Note that the problem of building a single minimum-cost network such that every pair of nodes in a given demand graph is connected

in the network is exactly the generalized Steiner network problem, for which polynomial-time constant-factor approximations exist.

Multicommodity facility location. Another stream of work has addressed the extension of facility location problems to reach clients with additional restrictions on the facility opening costs, to reach facilities more robustly [78], or with the addition of services that facilities open to satisfy the clients with various cost functions governing the installation of services and facilities [74, 76]. The work in [68] arising from publisher-subscriber mechanisms is most closely related to our work, and rather than use a network from each publisher, models the publisher as a commodity that can be supplied at various nodes in the network by installing "facilities" of appropriate costs; the subscribers build minimum-cost networks to reach these facility installations of the appropriate publishers.

Access Networks and Connected Facility Location. Our algorithms for the complete DON problem are connected to to the access network design and facility location problems. In a version of the Access Network Design problem [73], we are given an undirected graph, a root node and non-negative metric costs on the edges, along with a subset of terminal nodes. The goal is to design a backbone network in the form of a tree or tour which is built with higher speed and higher quality cables, while the terminals access the backbone using direct access edges. Thus the overall network is a backbone rooted Steiner tree (or tour), with access networks that are stars arising from the terminals and ending at the nodes of the backbone. We are given a cost multiplier μ that denotes the cost overhead factor for the backbone compared to the access network and the objective to be minimized is the total cost $\mu \cdot c(\text{backbone network } T) + \sum_{stars\ s} c(s)$. A $\rho_{AN} = 3+2\sqrt{2}$-approximation is presented for this problem [73] when the backbone is a ring and the access network is a star. Subsequent work [19, 43, 77] present constant-factor approximations for other generalizations and variants of this problem as well using LP rounding and primal-dual methods. The current best approximation factor for the CFL generalization is $\rho_{CFL} \leq 4$ [19], which also extends to tree-star access network design, i.e. $\rho_{AN} \leq \rho_{CFL} \leq 4$.

Virtual Private Network. The DON problems are also closely related to the VPN and asymmetric VPN problems. The VPN problem can be solved exactly [41], while the asymmetric VPN problem is NP-Hard but has a constant approximation [75]. The VPN problems differ from DON as each VPN problem instance seeks only one network, while a DON instance builds multiple networks. Nevertheless, we are able to decompose an approximate solution for asymmetric VPN into multiple networks, and obtain a useful approximation for the complete tree-tree DON problem (Section 4.4). As we establish in Section 4.3.1, however, a more direct approach yields a much better approximation factor.

4.2 DON with General Demands

In this section, we consider approximation algorithms for the general DON problem. We present in Subsection 4.2.1 an $\Omega(\log \log n)$ integrality gap for a natural LP relaxation of the tree-DON problem (with general demands). We present a constant approximation algorithm for tree-star DON on tree metrics and establish Theorem 4.2.2.

4.2.1 Integrality Gap for Tree-Tree DON

In this section, we show that a natural LP formulation for the tree-tree DON problem has super-constant integrality gap. We note that the same lower bound on integrality gap extends to the appropriate LP for general DON. A natural integer program, $\mathbf{IP_{BT}}$ for the tree-tree DON problem is as follows (all variables are $0-1$): we let $r \in P \cup S$ denote a publisher or subscriber node that serves as the root of its tree N_r; z_e^r is an indicator variable that is 1 iff edge $e \in E(G)$ is in tree N_r; y_h^r is an indicator variable that is 1 iff vertex h is in tree N_r; $x_h^{r,s}$ is an indicator variable that is 1 iff vertex h is in both trees, N_r and N_s; $\mathcal{C}$ will refer to a cut which is a subset of vertices of $V(G)$ and $E(\mathcal{C})$ will denote the edges between $\mathcal{C}$ and its complement $V(G) \setminus \mathcal{C}$. The integer program $\mathbf{IP_{BT}}$ for the tree-tree DON has the following nontrivial constraints.

$$\begin{aligned}
\min \sum_{r \in V(D), e \in E(G)} & c_e z_e^r \\
\sum_{e \in E(\mathcal{C})} z_e^r & \geq y_h^r \quad \forall \mathcal{C}, r \in \mathcal{C}, h \in V(G) \setminus \mathcal{C} \\
x_h^{r,s} & \leq y_h^r \quad \forall r, s \in V(D), h \in V(G) \\
\sum_{h \in V(G)} x_h^{r,s} & \geq 1 \quad \forall (r,s) \in E(D)
\end{aligned}$$

$\mathbf{IP_{BT}}$

The first set of cut covering constraints enforce that the tree rooted at r is connected with all nodes h for which y_h^r is set to one. The third set enforces all pairs of terminals r, s in the demand graph must meet in some hub vertex h, while the second set enforces that if a node h is used as a hub for a pair, it is required to occur in both these trees. Relaxing the above integer program by allowing the variables to take values in $[0, 1]$ gives us the natural linear program $\mathbf{LP_{BT}}$. Observe that the feasible integral points of the linear program are exactly the solutions to the integer program.

Theorem 4.2.1. For every sufficiently large n, there exist instances of tree-tree DON with $n = |V(G)|$ for which $\mathbf{LP_{BT}}$ has an $\Omega(\log \log n)$ integrality ratio.

Proof. Recall that the integrality ratio of a (minimizing) linear program is the minimum ratio between any feasible integral point and the optimum fractional solution. Our proof will proceed by a reduction from a linear program for the Group Steiner Tree (GST) problem.

Given a tree T with edge costs and a collection of groups of leaves, the Group Steiner Tree problem is to find a minimum cost subtree such that at least one vertex from every group is connected to the root. In [45] it was shown that a natural linear program for the GST problem has an $\Omega(\log^2 n)$ integrality ratio even when the input metric costs c arise from an underlying tree. Similar to the linear program for the tree-tree DON problem we present the linear program $\mathbf{LP_{GST}}$ as the relaxation of an integer program $\mathbf{IP_{GST}}$ with $0-1$ variables.

$$\min \sum_{e \in E(T)} x_e c_e$$
$$g_i \cap \mathcal{C} = \phi \quad \forall \mathcal{C}, r \in \mathcal{C}$$
$$\sum_{e \in E(\mathcal{C})} x_e \geq 1$$

$\mathbf{IP_{GST}}$

Here is an intuitive explanation of what the variables in $\mathbf{IP_{GST}}$ represent: x_e is an indicator variable that is 1 iff edge $e \in E(T)$ is in the solution subtree; $g_i, 1 \leq i \leq k$ are the k groups and $\mathcal{C}$ is a subset of vertices of $V(G)$ referring to a cut and $E(\mathcal{C})$ will denote the edges between $\mathcal{C}$ and its complement $V(G) \setminus \mathcal{C}$. The main cut covering constraints enforce that each group is connected to the root node r.

As stated before [45] show that $\mathbf{LP_{GST}}$ has an integrality ratio of $\Omega(\log^2 n)$ even on tree metrics when k, the number of groups, is $\Omega(n)$ where $n = |V(T)|$.

Given an instance, T_{GST} of $\mathbf{LP_{GST}}$ with $n = V(T)$ vertices and k groups, we transform it into an instance, GD_{BT} of $\mathbf{LP_{BT}}$ with $N = 2^k n = |V(G)|$ vertices in the host graph such that

- corresponding to every fractional solution, of value f_{GST}, of $\mathbf{LP_{GST}}$ there is a fractional solution of value $f_{BT} = 2^k f_{GST}$ to $\mathbf{LP_{BT}}$, and
- corresponding to every feasible integral point, of value I_{BT}, of $\mathbf{LP_{BT}}$ there is a feasible integral point of value $I_{GST} = \frac{I_{BT}(1+\log k)}{2^k}$ to $\mathbf{LP_BT}$.

The transformation is intuitive: we take a "graph product" of the Group Steiner Tree instance with a hypercube of dimension k, where k is the number of groups. In more detail, given the tree T for the Group Steiner Tree instance, the demand graph D is a hypercube of dimension k, with the ith dimension of the hypercube being associated with group g_i. The host graph G has 2^k copies of T, one for each vertex of the demand hypercube (where the demand edges go between the roots of the corresponding trees). For every edge (r, s) in D in the ith dimension we connect pairwise with zero-cost edges the leaves in group g_i of the copy of T corresponding to r, with the leaves in group g_i of the copy of T corresponding to s.

It is easy to see that $f_{BT} = 2^k f_{GST}$ for the above transformation - observe that replicating the fractional solution to $\mathbf{LP_{GST}}$ in each of the 2^k copies of T is a valid fractional solution to $\mathbf{LP_{BT}}$.

For the other direction, i.e., to see $I_{GST} = \frac{I_{BT}(1+\log k)}{2^k}$ first observe that for edge (r, s) in dimension i of the demand hypercube, at least one of the trees corresponding to r or s must cross dimension i and the only way to cross dimension i is along a 0-cost edge connecting two corresponding group g_i leaves. Now note that any tree T' in an integral solution to $\mathbf{LP_{BT}}$ can be transformed into a subtree of T by keeping an edge in T if T' contains the corresponding edge in any copy of T in G. Let the subtree of T so obtained be called the *retract* of T'. It is easy to see that if T' ever crosses dimension j then a leaf in group g_j is connected to the root of T in its retract and that the cost of a retract is never more than the cost of the original T'. By our earlier observation

for any edge (r, s) in dimension i at least one of the two retracts, that of the tree corresponding to r or corresponding to s must connect a node in group g_i to the root. Hence if we select a node in D at random and take its retract then any given group is connected with probability at least $1/2$ and it has expected cost $\frac{I_{BT}}{2^k}$. Thus if we take the union of $1 + \log k$ retracts chosen uniformly at random then the resulting subgraph of T has expected cost $\frac{I_{BT} \log k}{2^k}$ and the probability any given group is not connected to the root is less than $\frac{1}{k}$. Since there are k groups this means there exists a subgraph of T, connecting the root to every group, of cost at most $\frac{I_{BT}(1+\log k)}{2^k}$, i.e., $I_{GST} = \frac{I_{BT}(1+\log k)}{2^k}$.

From $f_{BT} \leq 2^k f_{GST}$ and $I_{GST} \leq \frac{I_{BT}(1+\log k)}{2^k}$ it follows that $\frac{I_{BT}}{f_{BT}} \geq \frac{I_{GST}}{f_{GST}}(1 + \log k)$. By [45], when $k = \theta(n)$ we have that $\frac{I_{GST}}{f_{GST}} = \Omega(\log^2 n)$ from which it follows that $\frac{I_{BT}}{f_{BT}} = \Omega(\frac{\log^2 n}{\log k}) = \Omega(\log n)$ but the size of the transformed instance is $N = 2^k n$, i.e., $n = \Omega(\frac{\log N}{\log \log N})$. In other words, the integrality gap $\frac{I_{BT}}{f_{TT}} = \Omega(\log n) = \Omega(\log(\frac{\log N}{\log \log N})) = \Omega(\log \log N)$ □

4.2.2 A logarithmic approximation for general DON

We next show that the general DON problem can be approximated to within an $O(\log n)$ factor in polynomial time. As discussed in Section 4.1, the general DON problem can be split into three problems: tree-tree DON, star-star DON, and tree-star DON. In previous work, $O(\log n)$-approximation algorithms have been developed for tree-tree DON and star-star DON [10]. We now present an $O(\log n)$-approximation for tree-star DON, implying an $O(\log n)$-approximation for general DON.

Our $O(\log n)$-approximation for tree-star DON is obtained by deriving a constant-factor approximation for the special case of tree metrics, and invoking the standard reduction from general metrics to tree metrics [25]. Our constant-factor approximation algorithm, which rounds an LP relaxation, essentially generalizes a result of [68] on multicommodity facility location from a uniform facility cost case to the case where the facility costs form a tree metric.

Theorem 4.2.2. The tree-star DON problem with general demands on tree metrics can be approximated to within a constant factor in polynomial time. This implies an $O(\log n)$-approximation algorithm for general DON on general metrics.

We first present a linear programming relaxation for the problem. Let T denote the given tree which is our metric. For a publisher j and an edge e of T, let z_e^j represent the extent to which j's tree uses e. For a subscriber i and leaf node v, let y_v^i denote the extent to which i's star visits v. For leaf node u, subscriber i and publisher j such that (i, j) is in the demand graph, let $x_u^{i,j}$ denote the extent to which j meets i at u. Let $d(u, v)$ denote the distance between u and v under the tree metric; abusing notation somewhat, let $d(e)$ denote the distance between the two endpoints of the

edge e. Then, we have the following LP.

$$\begin{aligned}
&\text{minimize} && \textstyle\sum_{j,e} z_e^j d(e) + \sum_{i,u} y_u^i d(i,u) \\
&\text{subject to} \\
&&& z_e^j \geq \textstyle\sum_{u:e\in P_{ju}} x_u^{i,j} && \text{for all } i,\, j,\, e \\
&&& \textstyle\sum_u x_u^{i,j} \geq 1 && \text{for all } (i,j) \text{ in demand graph} \\
&&& y_u^i \geq x_u^{i,j} && \text{for all } i,\, j,\, u \\
&&& x_u^{i,j}, y_u^i, z_e^j \geq 0 && \text{for all } i,\, j,\, u,\, e
\end{aligned}$$

We now present our algorithm. We introduce some useful notation first. Let Y_v^i denote the sum of y_w^i, over all leaves w in the subtree rooted at v. Similarly, let $X_v^{i,j}$ denote the sum of $x_w^{i,j}$, over all leaves w in the subtree rooted at v.

1. Solve the above LP to obtain a fractional solution (x, y, z).
2. For every subscriber i:
   - For every node v such that $Y_v^i \geq 1/3$ and there is no child c of v such that $Y_c^i \geq 2/3$: we mark node v.
   - For each marked node v such that no ancestor of v is marked, we add v to $\sigma(v)$; we refer to v as a type-C hub for i.
   - For every node v such that (a) there is no ancestor of v that is a type-C hub for i, and (b) there are two children c_1 and c_2 of v such that $Y_{c_1}^i \geq 1/3$ and $Y_{c_2}^i \geq 1/3$, we add v to $\sigma(v)$; we refer to v as a type-A hub for i.
   - For every node v that is an ancestor of a type-C hub, we define W_v^i to be sum, over every child c of v that is not an ancestor of a type-C hub, of Y_c^i.
   - For every path p from the root or a type-A hub node to a descendant type-A hub or type-C hub node: we divide p into minimal contiguous segments such that the sum of W_v^i, over all v in the segment, is at least $1/3$; for each such segment, we create a type-B hub for i at the lowest node in the segment.
   - The star network for i connects i to each type-A, -B, and -C hub.
3. For every publisher j, the tree network consists of all edges e such that $z_e^j \geq 1/3$.

We prove Theorem 4.2.2 by establishing the following two lemmas.

Lemma 4.2.3. For any edge (i, j) in the demand graph, the tree of publisher j overlaps with the star of subscriber i at least one node.

Proof. Fix publisher j and subscriber i such that (i, j) is an edge in the demand graph. Consider some subtree rooted at a node r_0 such $X_{r_0}^{i,j}$ is at least $1/3$ in the LP solution, while for any child c of r_0, $X_c^{i,j} < 1/3$. Suppose $(r_0, r_1, \ldots, r_f)$ denote the path from r_0 to the root of the tree.

We first show that if there is a type-C hub at a node r_k, then the tree of publisher j includes node r_k. By our algorithm's choice of locating type-C hubs, it follows that $Y_{r_{k-1}}^i < 2/3$. Therefore, publisher j meets subscriber i less than $2/3$ in the subtree rooted at r_{k-1}. We consider two cases. If j is in the subtree rooted at r_{k-1}, then for the edge $e = (r_{k-1}, r_k)$, $z_e^j \geq 1/3$. Otherwise, since

$X^{i,j}_{r_{k-1}} \geq X^{i,j}_{r_0} \geq 1/3$, again we have $z^j_e \geq 1/3$. Thus, in both cases, we ensure that the tree for publisher j contains r_k.

We next show that if there is a type-A or type-B hub for i at a node r_k and there are no hubs for i at any r_g, $0 \leq g < k$, then the tree for j must include r_k. Since there is no type-A hub at any r_g, $0 \leq g < k$, each r_g has at most one child that has a descendant with a type-C hub; if there were two such children, then r_g would have a type-A hub. Furthermore, there must be a type-C hub in the subtree rooted at r_0; if not, then the first ancestor of r_0 to have a hub would have a type-C hub, which would contradict our assumption. So suppose there is a type-C hub in the subtree rooted at r_0, say under the child r_{-1} of r_0. Then, it must be the case that the sum of $W^i_{r_g}$, over $0 \leq g < k$, is at most $1/3$, since otherwise we would have a type-B hub at r_g. Furthermore, by the definition of r_0, $X^{i,j}_{r_{-1}} < 1/3$. This implies that j meets i to an extent of $1/3$ outside the subtree rooted at r_{k-1} and at least $1/3$ inside the subtree rooted at r_{k-1}. Thus, regardless of where j is located for edge $e = (r_{k-1}, r_k)$, we will have $z^j_e \geq 1/3$, ensuring that the tree for publisher j contains r_k. □

Lemma 4.2.4. The total cost of the tree and star networks is at most a constant factor times the LP optimal.

Proof. An edge e is added to the tree of publisher j exactly when $z^j_e \geq 1/3$. Therefore, the cost of the tree network of j is within the cost for j in the LP.

We next consider the costs of the subscriber stars. There are three parts to it. First is the distance to the type-A hubs. If a type-A hub for i is created at a node r, then there exist two children c_1 and c_2 of r such that $Y^i_{c_1}$ and $Y^i_{c_2}$ are both at least $1/3$. Clearly, i is either not in the subtree rooted at c_1 or not in the subtree rooted at c_2. In either case, the cost for i in the LP solution for reaching the fractional hubs in one of c_1 or c_2 is at least $d(i, r)/3$. Adding this over all the type-A hubs yields a cost that is at most 3 times the LP cost for i.

If a type-C hub is created at a node r, then we consider two cases: the LP cost associated with the fractional hubs under the subtree at r is at least $d(i, r)/3$.

If a type-B hub is created at a node r, then consider the sequence of ancestors a of r, whose W^i_a add up to $1/3$. The cost of i reaching the fractional hubs in the LP that contribute to these W^i_a is at least $d(i, r)/3$.

The fractional hubs against which we have charged the type-C and type-B hubs are different, so the cost for the type-B and type-C hubs is at most 3 times the LP cost for i, yielding an $O(1)$-approximation for the overall total cost. □

4.3 DON with Complete Demands

In this section, we present constant factor approximation algorithms for the DON problem when the demand graph is complete. We obtain this result by deriving constant-factor approximations for the three variants—tree-tree, star-star and tree-star— in the following subsections.

4.3.1 Complete tree-tree DON

The complete tree-tree DON problem has an interesting connection to the asymmetric VPN problem [75], which we can exploit to obtain a constant-factor approximation. As we discuss this reduction and explain in Section 4.4, however, the approximation we obtain using the best known asymmetric VPN algorithm is 49.84. Here, we present a more direct approximation algorithm for which we are able to establish a much better approximation factor.

Theorem 4.3.1. There is a $4\rho_{TS}$-approximation algorithm for complete tree-tree DON, where ρ_{TS} is the best factor for the tree-star access network design problem.

In the rest of this subsection, we give a proof of Theorem 4.3.1. Given N^*, an optimal solution, let us denote the publisher and subscriber networks by $P_1, P_2, \ldots, P_k$ and $S_1, S_2, \ldots, S_l$ where we index the nodes so that we have $c(P_1) \leq c(P_2) \leq \cdots \leq c(P_k)$ and $c(S_1) \leq c(S_2) \leq \cdots \leq c(S_l)$. Let c^*_P and c^*_S denote the total cost of the publisher and subscriber trees. Let s_j be the subscriber whose network is S_j and let p_i be the publisher whose network is P_i. Note that feasibility requires that $P_i \cap S_j \neq \emptyset$ for all i, j. Let us also assume without loss of generality that $c(P_1) \leq c(S_1)$

The key transformation of the optimal solution is a reconfiguration of the subscriber networks where we replace each tree S_j for $j \neq 1$ by the direct edge from subscriber node j to subscriber node 1 concatenated with the subscriber tree S_1. In other words, we set $S'_j = \{(s_j, s_1)\} \cup S_1$ for every subscriber $s_j \neq s_1$. Let us assign $S'_j = S_j$.

Note that the modified subscriber trees are still feasible since the original subscriber tree S_1 intersects every publisher tree. We now bound the cost of the additional edge from subscriber j to the subscriber 1, the root of S_1.

Lemma 4.3.2. For every subscriber $j \neq 1$, we have $c_{i1} \leq 3c(S_j)$.

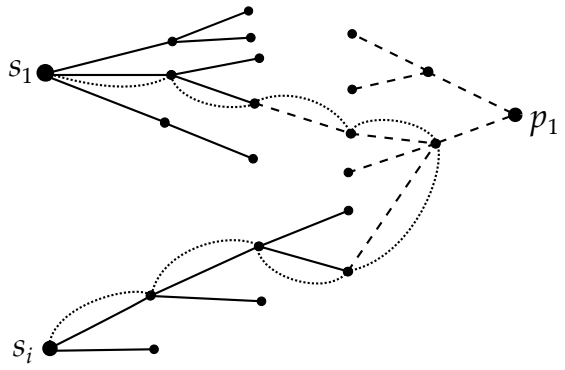


Figure 4.1: The solid lines show the trees S_i and S_1 and the dashed lines show the tree P_1. The dotted line here is the path from s_i to s_1 through the trees S_i, P_1, and S_1.

Proof. To see this, note that by taking the path from j in S_j to its intersection with P_1 and following it to the intersection of P_1 and S_1 and continuing along S_1 to the subscriber node 1, we have found a path from j to 1 of cost no more than the sum of the costs of S_j, P_1 and S_1. However, since $c(S_j) \geq c(S_1) \geq c(P_1)$, the length of this path is at most $3c(S_j)$. □

Given that every subscriber contains the tree S_1, it is particularly simple to design the publisher network P_i' (for publisher p_i) that needs to reach this tree: it will simply be a direct edge that represents the shortest path from the publisher p_i to the tree S_1. The union of all such direct edges gives a collection of stars that end at the subscriber tree S_1. Furthermore since the subscriber tree S_1 is going to be used by every subscriber node, its cost must be counted $|S| = l$ times in the objective.

The resulting problem of finding the best tree for S_1 is exactly the tree-star access network design problem [73] with the root being subscriber 1, the multiplier $M = |S|$ and the terminals being $R = P$, the publisher nodes. Given an optimal solution N^* for the complete tree-tree DON problem, we thus have a solution to the tree-star access network instance of cost at most $c(P^*) + |S| \cdot c(S_1)$. We thus have the following lemma.

Lemma 4.3.3. For the correct choice of the subscriber node 1 as the root with multiplier $|S|$ and terminals P, there is a solution to the tree-star access network design problem of cost at most $c^*_P + |S| \cdot c(S_1)$. □

Proof of Theorem 4.3.1. The approximation algorithm tries every subscriber node as the root of the tree-star access network problem formulated above. By adding the direct edge from each other subscriber to this root, and extending the backbone tree with each such edge, we get a solution to the complete tree-tree DON problem. The algorithm keeps the solution of smallest total cost among all choices of the root subscriber node. The total cost of the solution is the sum of the cost of the tree-star access network design problem and the sum of the costs of the direct edges from the subscribers to the root. By Lemma 4.3.2, the latter cost is no more than three times the cost of the tree (with the multiplier of —S—) in the solution to the tree-star access network design problem. By 4.3.3 and the ρ_{TS}-approximation factor for the tree-star access network design problem, we thus obtain a total cost of at most $4\rho_{TS}(c^*_P + |S| \cdot c(S_1))$ which is at most $4\rho_{TS}$ times the cost of N^*. □

It is not hard to extend the above methods to the case when the input terminals are partitioned into more than two subsets, say $R = P_1 \cup P_2 \cup \ldots \cup P_k$ and the demand graph is the complete k-partite graph between these k subsets. By considering the partition that has the cheapest tree network in the optimal solution to be in P_1, the above argument can be adapted to give a constant-factor approximation. We omit the details in this extended abstract.

4.3.2 Complete star-star DON

In this section, we present a constant-factor approximation for complete star-star DON.

Theorem 4.3.4. There is a 4α-algorithm for complete star-star DON, where α is the best approximation achievable for metric uncapacitated facility location.

Our algorithm and the proof of Theorem 4.3.4 are based on an argument that there exists a constant-factor approximate solution that has a special structure; our algorithm then computes a constant-factor approximate solution with this special structure.

Given a solution where the publisher network is $P_1, P_2, \ldots, P_k$ and subscriber network is $S_1, S_2, \ldots S_l$, let P_1 be the publisher network of smallest cost and S_1 be the subscriber network

of smallest cost without loss of generality. Also, let $\sigma(P_i)$ denote the set of nodes (which we refer to as hubs for P_i) in the star for the ith publisher. Likewise, let $\sigma(S_j)$ be the set of nodes in the star for the jth subscriber. Thus, we can refer to solutions using the maps defined by σ and denote the optimal one by σ^*. The next lemma shows a near-optimal solution with a very simple structure.

Lemma 4.3.5. There exists a solution σ such that $c(\sigma) \leq 4c(\sigma^*))$, and either $\sigma(S_i) = \sigma(S_j)$ for all pairs of subscriber networks S_i, S_j and $|\sigma(P_i)| = 1$ for each publisher network P_i or $\sigma(P_i) = \sigma(P_j)$ for all pairs of publisher networks P_i, P_j and $|\sigma(S_i)| = 1$ for each subscriber network S_i.

Proof. Without loss of generality, let $c(P_1) \leq c(S_1)$. Since each subscriber star intersects all publisher stars, we have $d(s_i, s_1) \leq c(S_i) + c(P_1) + c(S_1) \leq 3c(S_i)$. Let C_1 denote the set of publishers that share any hub with p_1. Let p_2 denote the least-cost publisher not in C_1. Let C_2 be the set of all publishers not in C_1 that share any hub with p_2. In general, let p_{j+1} be the least-cost publisher not in $\bigcup_{1 \leq i \leq j} C_i$. Let C_{j+1} denote the set of all publishers not in $\bigcup_{1 \leq i \leq j} C_i$ that share any hub with p_{j+1}. Let h_j denote any hub in $\sigma^*(s_1) \cap \sigma^*(p_j)$.

Let p'_j be an arbitrary publisher in C_j. We first obtain the following equation $d(p'_j, p_j) \leq 2c(P_{j'})$ (owing to a shared hub and the fact that $c(P_j) \leq c(P_{j'})$). By construction, for any two distinct p_i and p_j, we have $\sigma^*(p_i) \cap \sigma^*(p_j) = \emptyset$; i.e., p_i and p_j do not share any hubs. Note that this may not be true of all pairs of publishers in $C_i \times C_j$.

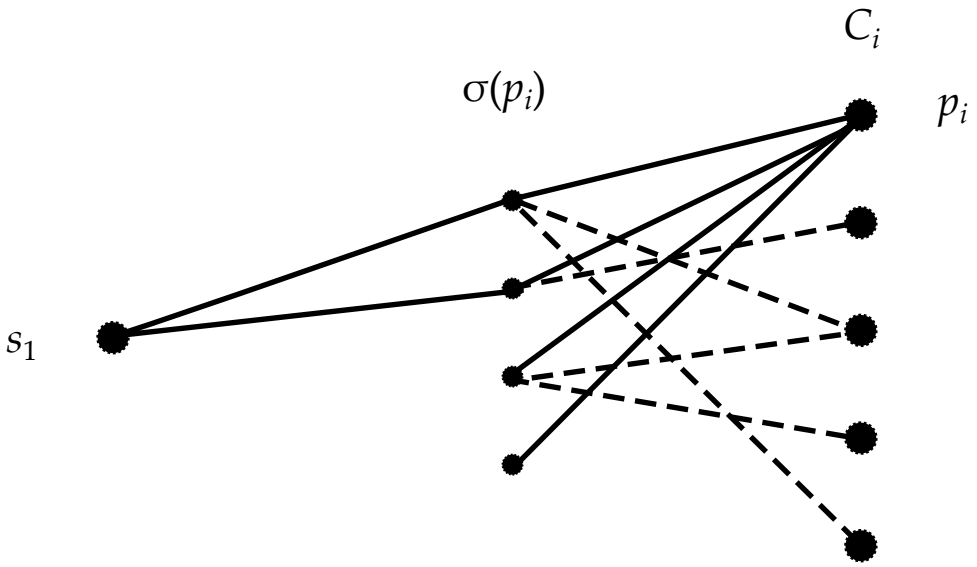


Figure 4.2: Here is an example of a P_i. We have shown all it's hubs, $\sigma(p_i)$. C_i consists of all those p_j's not in a previous C_k connecting to one of the hubs. Here h_i can be chosen to be either of the top two hubs in $\sigma(P_i)$.

We now consider two cases. In the first case when $c(P_j) \leq d(s_1, h_j)$, we have all subscribers meet all the publishers in cluster C_j at p_j. Consider any subscriber s_i. It meets p_j at some hub, say h^i_j. Its increase in cost for meeting p_j now is at most $c_{\sigma^*}(p_j) \leq d(s_1, h_j)$, which equals one leg of

s_1's star. Since two different p_j's do not share any hubs, the h_j^i's (for a given i) are all different. Hence, the total increase in cost for s_i is at most $\sum_j d(s_1, h_j)$, which is at most $c(S_1)$.

If $c(P_j) > d(s_1, h_j)$, then we will have all publishers in C_j, go to s_1. Fix a publisher p'_j in C_j. Its total cost is at most $d(p'_j, p_j) + c(P_j) + d(s_1, h_j) \leq d(p'_j, p_j) + 2c(P_j) \leq 4c(P'_j)$. All the subscribers also go to s_1 to handle this case. We have $d(s_i, s_1) \leq c(S_i) + c(P_1) + c(S_1) \leq 3c(S_i)$.

So overall, we obtain a blowup of at most 4 in the cost for each publisher and each subscriber. We have proved that there exists a solution of cost at most $4 \cdot OPT$ in which every subscriber's star connects to exactly the same set of hubs and every publisher's star is just a line to one of the hubs. □

Proof of Theorem 4.3.4. Using Lemma 4.3.5, we now give a polynomial-time 4α-algorithm where α is the best approximation achievable for the uncapacitated facility location problem.

Our algorithm considers all possible choices for S_1, a linear number (where by symmetry S_1 could be on either side). For a given choice of S_1, we formulate an uncapacitated facility location problem, with the set of publishers as the clients, and the potential facility locations being the publishers and S_1. The cost of opening a facility at any of these nodes is the sum of the distances of all the subscribers to that node. Given a solution to this facility location problem, we obtain a solution to the complete star-star DON problem as follows: each publisher's star is a singleton edge to the facility it is assigned to; each subscriber's star consists of edges to all the open facilities.

We solve all the linear number of facility location problems, and then the corresponding problems with the roles of subscribers and publishers reversed, and take the best solution. This yields the desired approximation. □

4.3.3 Complete tree-star DON

We now present a constant factor approximation for complete tree-star DON. Without loss of generality, let us suppose that the publishers will build trees, and the subscribers will build stars. The main idea is to show that either the appropriately defined complete star-star DON solution or complete tree-tree DON solution is within a constant factor of optimal.

Let N^* be an optimal solution. Let the trees be indexed $P_1, P_2, \ldots P_k$ and the stars $S_1, \ldots S_\ell$ such that $c(P_1) \leq \cdots \leq c(P_k)$ and $c(S_1) \leq \cdots \leq c(S_k)$.

First consider the case where $c(P_1) \geq c(S_1)$. Note that every P_i and S_j must have a non-empty intersection. Now for every tree P_j we can redirect it to P_1 and then make a copy of P_1. So we will let: $P'_j = \{(p_j, p_1)\} \cup P_j$.

This solution is feasible because P_1 must intersect all the stars. These additions to the solution cost at most $3c(N^*)$, as seen in lemma 4.3.2. Now all the stars can simply take an edge which is the shortest edge to the tree.

The approximation algorithm from this point follows the tree-tree case exactly. In this case, we get that the final solution has cost at most $4\rho_{TS}c(N^*)$. Where ρ_{TS} is the best constant approximation for the tree-star access problem.

Next consider the case that $c(S_1) \geq c(P_1)$. We will now choose p_i's in a similar fashion to the complete star-star DON problem. Let p_1 be the publisher with the smallest cost tree. Let C_1 be all the publishers whose trees meet p_1's tree. Now let p_2 be the smallest tree which does not

intersect p_1's tree. Let C_2 be all the publishers not in C_1 who meet p_2's tree. Likewise p_{j+1} will be the smallest tree not in $\cup_{1 \leq i \leq j} C_i$. Let C_{j+1} be all the publishers which intersect p_{j+1}'s tree not in $\cup_{1 \leq i \leq j} C_i$.

Now from hereon, the proof follows that for the complete star-star DON case. Hence we have a solution within a constant factor of optimal where all the stars go to s_1 (the subscriber with star S_1), and some of the publishers; P_{open}. Each tree goes to the nearest node in $S_1 \cup P_{open}$. This establishes the following lemma.

Lemma 4.3.6. The complete tree-star DON has an $O(1)$-approximate solution in which either all the subscribers go to some hubs and each tree goes to the nearest hub among a set of one subscriber and some publishers, or where all the publisher trees are identical and all the subscribers go to the closest node in that tree. □

For solving the complete tree-star DON problem, we apply our constant-factor approximation algorithm for the complete tree-tree DON instance, together with our constant-factor algorithm for the complete star-star DON instance, and take the better of the two. This completes our argument showing that complete tree-star DON can be approximated to within a constant factor.

4.4 Alternate Algorithm for Complete Tree-Tree DON using VPNs

We use results from the asymmetric VPN problem [41, 75] and show how it gives an approximation for complete tree-tree DON. First we introduce the VPN problem. Given a graph G, with edge costs c, and marginals for each vertex, the VPN problem is to build a network of minimum cost such that for any set of pairwise demands which obey the marginals, the flow can be routed on our network. A set of pairwise demands obeys the marginals if the demands a vertex is involved in does not exceed its marginal. One crucial distinction between VPN and the DON problems is that while the VPN problem seeks the design of a single network, DON problems seek networks for every node involved.

Now we define asymmetric VPN. Here the flows are directed, and each vertex has two marginals, one for how much can flow out of the node, and one for how much can flow into the node. We restrict to the case where the terminals allow 1 flow out of the node and no flow in, *sources*, or they allow 1 flow into the node, and no flow out, *sinks*.

It turns out that for asymmetric VPN, there is always a tree solution which is within a constant factor of an optimal solution [75]. We now use this tree solution to get a solution for complete tree-tree DON.

Lemma 4.4.1. Given a complete tree-tree DON problem, consider an asymmetric VPN problem with the same input as the DON problem, with the subscribers as sources, and the publishers as sinks. Then, any tree solution for the asymmetric VPN problem can be transformed into a solution of the same cost for the complete tree-tree DON problem.

Proof. Let T be the tree which is a solution to asymmetric VPN. Since our solution to asymmetric VPN is a tree which is adjacent to all the publishers and subscribers, then every edge in the tree induces a partition of the terminal nodes.

Consider any edge $e \in T$; we decide which trees use e. Let S_1, P_1 be the subscribers and publishers respectively on one side of the partition; likewise let S_2, P_2 be the remaining subscribers and publishers respectively. Now let $a = \min(|S_1|, |P_2|)$ and $b = \min(|P_1|, |S_2|)$. Now a valid demand matrix would be to require a unit flow from a elements of S_1 to a elements of P_2 and to require a unit flow from b elements of S_2 to b elements of P_1. These $a + b$ flows must all cross e since T is a tree, therefore e has multiplicity at least $a + b$.

Now if $|S_1| \leq |P_2|$, then we assign e to be in the trees for the elements of S_1, otherwise e is in the trees for the elements of P_2. Likewise if $|S_2| \leq |P_1|$ we assign e to be in the trees for the elements of S_2, otherwise e is in the trees for the elements of P_1. The number of times we use e is

$$\min(|S_1|, |P_2|) + \min(|S_2|, |P_1|) = a + b$$

So, we don't overuse e.

We next need to show that the edges assigned to a node form a tree. Since the original structure was a tree, we only need to show that the edges assigned to a terminal t are connected. Without loss of generality, suppose that a copy of e was assigned to be in T_s for $s \in S_1$. Let Q be the path in the tree T from e to s. Let $e' \in Q$. Let V' be the vertices on the same side as s of the partition formed by removing e' from T. We know that $S \cap V' \subseteq S_1$. Likewise we know that $P_2 \subseteq P \cap V'^C$. Since $|S_1| \leq |P_2|$ (because $e \in T_s$), then we know $|S \cap V'| \leq |P \cap V'^C|$. So, e' is assigned to be in the tree for nodes in $S \cap V'$. Therefore we have that $Q \subseteq T_s$. Therefore e is connected to s. Hence the graphs formed by our assignment scheme are connected.

Lastly, we must show that for every $s \in S$ and $p \in P$ then T_s and T_p intersect. Consider $s \in S$ and $p \in P$. Let Q be the path in T from p to s and e be an edge in Q. When we look at the S_1, S_2, P_1, P_2 formed by removing e, then either $s \in S_1$ and $p \in P_2$, or $s \in S_2$ or $p \in P_1$. Without loss of generality, assume $s \in S_1$ and $p \in P_2$. Then e is assigned to be in either the tree for all elements of S_1 or for all elements in P_2. So e is in either T_s or T_p. Since T_s and T_p are connected subtrees of the same tree, and $Q \subseteq T_S \cup T_p$ then T_s and T_p meet at some vertex in Q. Therefore, all demands are satisfied and this is a valid solution to the complete tree-tree DON problem. □

This provides an 49.84 approximation algorithm for the complete tree-tree DON problem.

4.5 DON with complete Demands

4.5.1 Subscribers are also Publishers

For this section, we will begin by considering the case where $P = S$ and the set of demands is $P \times P$. We present the optimal solutions for the star-star case where the edge costs are metric and the tree-tree case. We define a *hub star* to be solution such that there is some node h such that the network N_p for each node is the smallest network containing h. In the star-star case each N_p will just be an edge in a hub star solution. In the tree-tree case each N_p will be a shortest path to h in a hub-star solution. The optimal solution for the star-star case with metric costs and the tree-tree case is the cheapest hub star. Clearly, computing the optimal solution in both of these cases becomes straightforward; simply try every possibility for the center of the hub star and take the cheapest one.

Theorem 4.5.1. There exists an optimal solution for complete tree-tree (star-star) DON with $P = S$ which is a hub star.

Proof. To show optimality, we will start with any feasible solution; $N = \cup_{p \in P} N_p$. On this network, we will construct a multicommodity flow with cost at most that of N. Then we will show that the flow we constructed is a convex combination of hub stars, and hence the cheapest hub star is cheaper than the flow.

For any solution $N = \cup_{p \in P} N_p$, let $c(p, q)$ be any node in both N_p and N_q. We will call $c(p, q)$ the connection between p and q. For simplicity , we will have $c(p, q) = c(q, p)$ and $c(p, p) = p$. Let $w_p(v)$ be the number of connections v makes to connect p to other nodes; this will be the weight of v with respect to N_p.

Now we will begin with the star-star case where the edge costs are metric. Let $h(p)$ be a node in N_p which has the largest weight with respect to N_p. We will call $h(p)$ the hub of p.

Now to set up the flow, first each node in P will have one unit of a commodity. For each pair, $p \neq q$ and $p, q \in P$, then they will send $1/n$ of their commodity to each other with $c(p, q)$ being the only intermediate node. Now each node in P has $1/n$ of each commodity at it. Every p will now push all the commodities it has to $h(p)$. So, the paths that this flow follows are of the form $p, c(p, q), q, h(q)$. Where the first edge it follows it uses the edge in N_p and the second two edges it follows are in N_q. If it is the case that $c(p, q) = h(q)$ we will allow this path to shortcut and just become $p, h(q)$. Now we will show that no edge in N has more than one unit of flow.

Consider an edge in N_p which goes to a vertex v which is not the hub of p. So, p will send $w_p(v)/n$ of it's commodity along this edge, and receive $1/n$ of $w_p(v)$ different commodities. The total flow on this edge is $2w_p(v)/n$. Since v is not the hub of p there is some node with weight at least as large as v. So, $w_p(v) \leq n/2$ and the flow on the edge pv is at most 1.

Now consider the edge going to $h(p)$ in N_p. If $c(p, q) = h(p)$, then p will send $1/n$ of it's commodity on edge $(p, h(p))$ to get to q, but q's commodity will not use any of the edges in N_p. If $c(p, q) \neq h(p)$, then p will use the edge $(p, h(p))$ to push $1/n$ of q's commodity to $h(p)$. Therefore for each $q \in P$, it will cause exactly $\frac{1}{n}$ flow on this edge. So, this edge will have flow exactly 1 on it.

Therefore the given flow is valid and has cost at most that of the original solution N. Each $h(p)$ receives $1/n$ flow from each node $q \in P$. The costs are metric, therefore the cost of the flow for these commodities must be at most $1/n$ of the cost to connect each of the $q \in P$ directly to $h(p)$. So, the flow has cost at most that of a convex combination of hub stars. Hence, the cheapest hub star has cost at most that of the flow. So, the cheapest hub star is at most the cost of any feasible solution. The cheapest hub star is always an optimal solution.

We will now consider the tree-tree case. We will assume that all the N_p in N are trees. If a given N_p is not a tree, then we can simply remove an edge in a cycle of N_p and reduce the cost while maintaining connectivity. The *centroid* of a tree is a node v such that removing the node creates subtrees each with weight at most $n/2$. We will redefine $h(p)$ to be a centroid of the tree N_p with respect to the weights w_p.

Now the flow we will create will be similar to the flow for the star-star case. Each node in P will have one unit of a commodity. Now for each $p, q \in P$ we will have p send $1/n$ of it's commodity to $c(p, q)$ via N_p and then from $c(p, q)$ to $h(q)$ via N_q.

Now consider any edge e in N_p. Consider removing this edge, it will create two subtrees T_1 and T_2. Let $w_p(T_i)$ denote $\sum_{v \in T_i} w_p(v)$. Without loss of generality let p be in T_1. So, e will have to send $w_p(T_2)/n$ of p's commodity for when the commodity goes from p out to it's connections. If $h(p) \in T_1$, then e will also have to handle $w_p(T_2)/n$ flow coming to $h(p)$. Since $h(p) \notin T_2$ then it must be the case that $w_p(T_2)$ is at most $n/2$ since $h(p)$ is the centroid. So, the total weight that e handles in this case is at most $2w_p(T_2)/n \leq 1$. If $h(p) \in T_2$ then e will have $w_p(T_1)/n$ flow coming into $h(p)$. We know that $w_p(T_1) + w_p(T_2)$ must account for the weight of all the vertices and must be exactly n. So, in this case e has a flow of exactly 1 on it.

Again we have that the flow is valid and has cost at most that of the original solution N. Each $h(p)$ receives $1/n$ flow from each node $q \in P$. The flow is at most a convex combination of hub stars again. Thus, a hub star is optimal for the tree-tree version where all demands are present.

In both of these cases if there are no zero cost edges, and the feasible solution has more than an edge or a path to $h(p)$ from p for any N_p than the flow has cost strictly less than that of the feasible solution, as there is some edge with flow strictly less than one. Therefore, when there are no zero cost edges then a hub star is the only form an optimal solution can have. □

There is an alternate proof of this theorem which follows via a reduction to the VPN problem and then using the result that an optimal VPN solution is a hub-star [41].

4.5.2 Publishers and Subscribers are disjoint

Now we will consider the case where P and S are disjoint, but both have exactly n vertices. We will work in the case where the demands are all of $P \times S$. Again we will be looking at the star-star case with metric edge costs, and the tree-tree case. We will show that the cheapest hub star solution is within $9/8$ of optimal in both these cases.

To show this, we will start with any feasible solution, and then create a flow which puts at most flow $9/8$ on each edge. This flow will again be a convex combination of hub stars.

For this part, we let $c(p, s)$ be defined for $p \in P$ and $s \in S$ as some node in N_p and N_s. We will again use $w_p(v)$ to be the number of connections that v makes for p.

Starting with the star-star case on metric costs, the flow will be similar to the previous part without a dedicated hub for each vertex. We will start with any feasible solution $N = (\cup_{p \in P} N_p) \cup (\cup_{s \in S} N_s)$. Again each $p \in P$ and each $s \in S$ starts with one unit of it's own commodity. Each $s \in S$ will start by sending $1/n$ of it's commodity to $c(p, s)$ via N_s for all $p \in P$. Now from here it will split it's commodity further. For each $v \in N_p$, s will send $w_p(v)/n^2$ from $c(p, s)$ to v using N_p. Likewise, each $p \in P$ will start by sending $1/n$ of it's commodity to $c(p, s)$ via N_p for all $s \in S$. Again further splitting the commodity, for each $v \in N_s$, p will send $w_s(v)/n^2$ from $c(p, s)$ to v using N_s.

Now consider any $p \in P$ and any edge (p, v) in N_p, let $x = w_p(v)/n$. We will look at the flow that this edge receives. It will get x flow of p's commodity going to v. Now $1 - x$ of the flow that v receives from other nodes will have to go into p on their way to other nodes in N_p. v initially receives x flow from nodes in S. So, the flow from v to p is of size $x(1 - x)$. Lastly, there will be $1 - x$ commodities coming to N_p via nodes besides v. We will need to get x of these to v. So, the flow of commodities from S going to v from elsewhere N_p will contribute $x(1 - x)$. So, the total

flow that this edge receives is

$$x + 2x(1 - x).$$

This quantity is maximized when $x = \frac{3}{4}$. So, the flow along this edge is at most $\frac{9}{8}$. By symmetry, all edges in any N_s also have flow at most $\frac{9}{8}$. Hence the cost of the flow is at most $\frac{9}{8}$ the cost of our original feasible solution.

Now for each $p \in P$ and $s \in S$, we have that $c(p, s)$ will receive $1/n^2$ of each of the $2n$ commodities from both P and S. Since the flow may take a path instead of an edge, we can shortcut and just take the direct edge because the edge costs are metric. So, the cost of the flow is again bounded below by a convex combination of hub stars. Hence, the cheapest hub star is within $\frac{9}{8}$ of optimal in this case.

Let $N = (\cup_{p \in P} N_p) \cup (\cup_{s \in S} N_s)$ be a feasible solution for the tree-tree case. Again we can assume that each N_p is a tree, as if is not a tree we can always remove an edge in a cycle to get a network of smaller cost that is still feasible. The flow is set up in exactly the same way as the star-star case on metric costs. We only need to be careful when examining the flow on each edge.

Consider any $p \in P$ and $e \in N_p$. Let T_1 and T_2 be the subtrees of N_p generated by removing e. Again let $w_p(T_i)$ be $\sum_{v \in T_i} w_p(v)$. Let $p \in T_1$ and let $x = w_p(T_2)/n$. The flow of p's commodity from p to the N_S will be x across this edge. Now for the x flow of commodities from S which reach N_p via T_2, e will have to support $1 - x$ of this x flow going from T_2 to T_1. For the $1 - x$ flow of commodities from S which reach N_p via T_1, e will have to support x of the $1 - x$ of these commodities which go from T_1 to T_2. Again we have the total flow on this edge is:

$$x + 2x(1 - x)$$

So, it is maximized when $x = \frac{3}{4}$ and is at most $\frac{9}{8}$. Likewise for every edge in a N_s it has flow at most $\frac{9}{8}$. Therefore the cost of the flow is at most $\frac{9}{8}$ that of the original solution N.

Now for each $p \in P$ and $s \in S$, we have that $c(p, s)$ will receive $1/n^2$ of each of the $2n$ commodities from both P and S. Since the flow may not take the shortest path, we can always take the shortest path without increasing costs. So, the cost of the flow is again bounded below by a convex combination of hub stars. Hence, the cheapest hub star is within $\frac{9}{8}$ of optimal in this case.

Unfortunately, a hub-star solution is only good when P and S are close in size. Without loss of generality, let $|P| > |S|$. With unequal P and S, then a hub star is only within $\frac{9}{8}\frac{|P|}{|S|}$ of optimal. This can be seen by using a similar flow, simply scaled to account that one side is larger than the other.

4.6 Conclusion

We gave constant factor approximations in the case where the demands were complete bipartite. In the case of the complete demand graph, then we showed that an optimal solution is a hub-star and can be found in polynomial time. For general demands, we addressed the tree-star case and gave an $O(\log n)$-approximation algorithm. Lastly, we gave an $\Omega(\log \log n)$ integrality gap on the natural LP relaxation.

Perhaps the most interesting set of questions arising from our work is the approximation status of the bipartite and more general DON problems with general demands. While the integrality

gap demonstrates that an LP rounding approach based on the natural formulation will not yield constant guarantees, we have not yet been able to convert these ideas into a hardness of constant-factor approximation proof.

Another avenue of open problems concern removing the metric requirement on the cost function: the tree-DON and tree-star-DON problems then become significantly different and harder (they generalize group Steiner problem, for instance) and require new ideas.

Finally, there exists $O(\log n)$-approximations for all three design metrics, but the techniques used are specific to each problem. One further avenue would be to find a unified $O(\log n)$-approximation for all three design types:star-star, tree-tree, tree-star.

Chapter 5

Tree Augmentation Problem

5.1 Introduction

We consider the *weighted tree augmentation problem (TAP)*: Given an undirected graph $G = (V, E)$ with non-negative weights c on the edges, and a spanning tree T, find a minimum cost subset of edges $A \subseteq E(G) \setminus E(T)$ such that $(V, E(T) \cup A)$ is two-edge-connected. Here $E(G)$ denotes the edges of G and $E(T)$ denotes the edges of T. A graph is *two-edge connected* if the removal of any edge does not disconnect the graph, i.e., it does not have any cut edges. Since cut edges are also sometimes called bridges, this problem has also been called *bridge connectivity augmentation* in prior work [31]. We will call the elements of $E(T)$ as (tree) edges and those of $E(G) \setminus E(T)$ as *links* for convenience.

While TAP is well studied in both the weighted and unweighted case [2, 11, 16, 28, 31, 54, 56, 71], it is NP-hard even when the tree has diameter 4 [31] or when the set of available links form a single cycle on the leaves of the tree T [13]. Weighted TAP remains one of the simplest network design problems without a better than 2-approximation. TAP can also be viewed as a covering problem. The cuts in a tree which have a single edge crossing them are exactly the cuts that must be covered.

A link ℓ is said to *cover* an edge e if the unique cycle of $\ell + T$ contains e. Here we use $\delta(e)$ for a tree edge e to denote the set of links which cover e. The natural covering linear programming relaxation for the problem, EDGE-LP, is a special instance of a set covering problem with one requirement (element) corresponding to each cut edge in the tree (Since the tree edges define shores that form a laminar family, this is also equivalent to a laminar cover problem [13]).

$$\min \sum_{\ell \in A} c_\ell x_\ell$$

$$x(\delta(e)) \geq 1 \qquad \forall e \in E(T) \tag{5.1}$$

$$x_\ell \geq 0 \qquad \forall \ell \in A \tag{5.2}$$

Fredrickson and Jájá showed that the integrality gap for EDGE-LP can not exceed 2 [31] and also studied the related problem of augmenting the tree to be two-node-connected (biconnectivity

versus bridge-connectivity augmentation) [32]. Cheriyan, Jordán, and Ravi, who studied half-integral solutions to EDGE-LP and proved an integrality gap of $\frac{4}{3}$ for such solutions, also conjectured that the overall integrality gap of EDGE-LP was at most $\frac{4}{3}$ [13]. However, Cheriyan et al. [14] demonstrated an instance for which the integrality gap of EDGE-LP is at least $3/2$.

We study the integrality gap of the EDGE-LP and its generalizations in this work. We first show that without loss of generality, we can focus our attention on binary trees where every node has degree 1 or 3 (and every link goes between a pair of leaves). By focusing on the internal nodes of degree 3, we can add a simple valid constraint. In particular, at any node of degree 3, since no link can cover all three edges which meet at this node, the total number of (integral) links which must cover its neighbors is at least 2. This gives one additional constraint per internal node that we can add to the EDGE-LP. The resulting LP, called the NODE-LP follows where we use $\delta_T(v)$ for a node v to refer to its three incident edges in the tree T.

$$\begin{aligned} \min \sum_{\ell \in E} c_\ell x_\ell & \\ x(\delta(e)) \geq 1 & \qquad \forall e \in T \\ x(\delta(e_1) \cup \delta(e_2) \cup \delta(e_3)) \geq 2 & \qquad \forall v \in T \text{ and } \delta_T(v) = \{e_1, e_2, e_3\} \\ x_\ell \geq 0 & \qquad \forall \ell \in E \end{aligned} \tag{5.3}$$

Fiorini et. al extended node constraints for all classes of odd subsets of tree edges as $\{0, \frac{1}{2}\}$-Chvátal-Gomory cuts of EDGE-LP to obtain new constraints on all odd sets of edges [28]. We call their extended linear program the ODD-LP. Since we will show that we can assume the tree is binary, every node has odd degree (1 or 3) in the input tree, so if $S \subseteq V$ is odd, then it follows that $\delta(S) \cap T$ is also odd. Using this observation, we can write the ODD-LP as follows. Recall that $\delta(S)$ for $S \subset V$ is the set of all edges and links with exactly one endpoint in S.

$$\begin{aligned} \min \sum_{\ell \in E} c_\ell x_\ell & \\ x(\delta(S)) + \sum_{e \in \delta(S) \cap T} x(\delta(e)) \geq |\delta(S) \cap T| + 1 & \qquad \forall S \subseteq V, |S| \text{ odd} \\ x_\ell \geq 0 & \qquad \forall \ell \in E \end{aligned} \tag{5.4}$$

In addition to the standard version, we also study the problem of 3TAP in which every tree edge in the final solution must be in a cycle of length 3 (instead of every tree edge being in a cycle of any length). This is a natural variant of TAP. While TAP models increasing the resilience of a tree network, 3TAP requires local resilience: i.e., in case of any edge failure, the overhead of implementing a rerouting protocol is not too high (3TAP solutions only need the identity of the midpoint of the alternate 2-path for every edge in the solution).

5.1.1 Related Work

Weighted TAP has several 2-approximation algorithms. The earliest proof of this result used methods that were tailored for this problem: Frederickson and JáJá [31] convert the problem into one

of finding a minimum weight arborescence in an appropriate directed graph: First, they root the given tree at an arbitrary node and direct it outwards; Links that go from a node to an ancestor are directed upward in the tree, while cross links are replaced by two links of the same weight going from each endpoint to their least common ancestor in the tree. After given the original tree edges directed downward weight zero, their method finds a minimum weight in-arborescence pointing to the root, which they argue is of cost at most twice the optimal weighted TAP solution for this instance (coming from the duplication of cross links). Khuller and Thurimella improved the runtime of this algorithm [54]. It is also worth noting that the directed instance when viewed as an undirected instance of TAP consists of all links going top-down in the tree (since cross links are replaced with two such links from their ends to their LCA). The EDGE-LP for all links going top-down in a tree is totally unimodular (see, e.g., Section 2 of [40]). Hence this version can be solved to optimality (providing an alternate to the use of the in-arborescence algorithm). Later, other 2-approximation algorithms have been devised for weighted TAP using other techniques such as the primal-dual method [71] and iterative rounding [50].

Special cases of weighted TAP has also been investigated. Cheriyan, Jordán and Ravi [13] developed a $\frac{4}{3}$-approximation for TAP when the optimal fractional solution is half-integral. Another special case of weighted TAP is when the tree has bounded depth. In this special case, Cohen and Nutov showed there exists a $(1 + \ln 2)$-approximation [16]. Recently, Adjiashvili [2] showed a 1.96-approximation for another special case of weighted TAP where all link weights are between 1 and some constant M by using a bundling type linear program. Building off this work, Fiorini et. al [28] generalized the constraints from [55] and combined them with the bundle constraints from [2] to propose the ODD-LP we described above and achieved a $\frac{3}{2} + \epsilon$ approximation for the same special case (when all the costs are between 1 and some constant M). Another recent paper by Nutov takes a subset of Adjiashvili's constraints and achieves a $\frac{12}{7} + \epsilon$ approximation when all the costs are between 1 and some constant M [67]. All of these techniques rely heavily on the bundle constraints that are focused on link weights being in a bounded range; hence they do not seem to be generalizable to the case of arbitrary weights. We believe the general problem requires a more polyhedral approach of the type we investigate.

Numerous papers attempted to reach a target $\frac{3}{2}$-approximation in the unweighted case of TAP when all links have the same weight. One paper by Kortsarz and Nutov [55] presents a new linear program with a 1.75-approximation for the unweighted case, in the hope that this linear program could help break the 2-approximation barrier for the weighted case. This LP used properties of an optimal solution for the unweighted case to add multiple new constraints; In retrospect, these additional constraints are all included in the ODD-LP. Two papers achieved a $\frac{3}{2}$-approximation for the unweighted case with very different approaches; one paper by Kortsarz and Nutov relies on a unique token giving argument [56]. The other paper by Cheriyan and Gao uses semi-definite programming [11, 12] to arrive at an initial fractional solution for which this integrality gap is proved. While both of these approaches are very different, they still heavily rely on the fact that all the links have the same weight.

5.1.2 Our Results

Our results gives new approaches to determine the integrality gap of weighted TAP: our methods provide constructive proofs of convex decompositions of given fractional solutions appropriately scaled into integer solutions.

1. We show that any instance of weighted TAP can be reduced to equivalent instances where the underlying tree is binary and all the links have their endpoints at leaves (Theorem 5.2.1 in Section 5.2). The simpler structure of input instances helps us in several of our proofs and may also be key in future approaches in settling the integrality gap of weighted TAP.
2. We give a simple new top-down coloring algorithm that gives a constructive proof of the integrality gap of 2 for EDGE-LP by providing a convex decomposition. Furthermore, if the minimum non-zero value in the solution for any link is α then we can achieve an improved $\frac{2}{1+\alpha}$-approximation (Theorem 5.3.1 in Section 5.3). This result generalizes the result of Cheriyan et al. [13]which we can recover by setting $\alpha = \frac{1}{2}$. Even more interestingly, this provides a new $\frac{3}{2}$-approximation when all nonzero values in the solution are at least $\frac{1}{3}$.
3. We provide a new conjecture on the ODD-LP (Conjecture 5.3.3) that says that every vertex solution to this LP has all large nonzero entries (greater than $\frac{1}{3}$) or there is a single very large valued entry (at least $\frac{2}{3}$). In the former case, we can use the previous theorem to get a $\frac{3}{2}$-approximation while in the latter, we can apply one step of iterative rounding [60], and reapply the conjecture to prove a $\frac{3}{2}$-approximation.
4. We provide a $\frac{3}{2}$-approximation for weighted TAP based on fractional solutions to NODE-LP with a particular structure. Let a *deficient* edge be an edge which gets covered to the extent less than $\frac{4}{3}$ by this fractional solution. In Section 5.4, we show that if the deficient edges for the NODE-LP form at most two paths in the tree, then we can extend our coloring construction to give a $\frac{3}{2}$-approximation.
5. Even though we provide improved approximations for specially structured extreme points of NODE-LP, we can show that such constraints do not strengthen EDGE-LP. In particular, in Section 5.5, we show how to transform any TAP instance to a slightly bigger one by a gadget expansion at every node so that any feasible solution to the EDGE-LP on the original instance is feasible to the NODE-LP in the expanded instance. Moreover, EDGE-LP has extreme points which violate our conjecture above, motivating a deeper study of ODD-LP for future work.
6. In Section 5.7, we provide a complete study of 3TAP in which every tree edge must be in a triangle in the final solution. Via a reduction from set cover, we show an $\Omega(\log n)$-inapproximability result and give a matching approximation algorithm. In the unweighted case, we show that any minimal solution gives a 4-approximation.

Our approach is a top-down coloring algorithm on the scaled fractional solution where each color class is a feasible solution. In particular, $\frac{3}{2}$ times the fractional solution is decomposed into a convex combination of integer solutions. This provides not only an approximation algorithm but also directly proves the integrality gaps for the corresponding covering LPs [9]. In addition, this technique of top-down coloring differs from all current $\frac{3}{2}$-approximation algorithms on un-

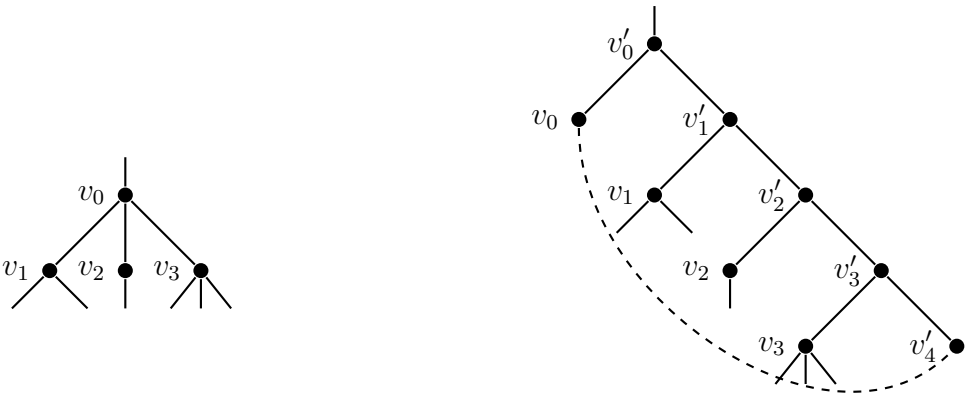


Figure 5.1: An example of a v_0 node with three children before and after the transformation.

weighted TAP and all current algorithms which achieve better than 2-approximations for special cases of weighted TAP. Since our methods decompose scaled fractional solutions, they also have the potential to extend to give tight integrality gap proofs - we propose some ideas for doing this in Section 5.5.

5.2 Problem Structure

In this section, we show that we can restrict our attention to only certain instances of weighted TAP. This structure restricts not only the structure of the links but also the structure of the tree itself.

Theorem 5.2.1. Any instance of weighted TAP (T, c, L) can be reduced to a corresponding instance of weighted TAP (T', c', L') of where the tree T' is binary, T' has at most three times as many nodes as T, and all the leaves in L' go between two leaves. In addition, every feasible solution to (T, c, L) provides a feasible solution to (T', c', L') of equal cost and vice versa.

The construction is a local operation performed on all the nodes of T in a top-down fashion. Let $v = v_0$ be a node in the tree with children $v_1, v_2, \ldots v_k$ (if v_0 is a leaf then no operation will be done). Let (T, c, L) be the initial tree, the transformation on v will give us a new instance (T_v, c_v, L_v). We will add dummy nodes v'_i for v and all its children and a dummy node v'_{k+1} for v. We remove the edges $X = \{v_0 v_i\}_i$ and add the edges $Y = \{v_i v'_i\}_i \cup \{v'_i v'_{i+1}\}_i$. We leave all the existing links at their corresponding nodes. The only link we add is a link called ℓ_v from v_0 to v'_{k+1} of cost 0. The new instance has changed as follows:

$$\begin{aligned} V(T_v) &= V(T) \cup \{v'_i : 0 \leq i \leq k+1\} \\ E(T_v) &= E(T) - X + Y \\ L_v &= L \cup \{v_0 v'_{k+1}\} \end{aligned}$$

Figure 5.1 gives an example of this transformation on a node with three children.

We will now show that performing this transformation on every non-leaf vertex of T produces an instance of TAP with a binary tree and leaf-to-leaf links with corresponding feasible solutions to the original problem.

Proof. First we observe that this transformation adds nodes $v'_0, v'_1, \dots v'_k$ all of degree 3, adds node v'_{k+1} of degree 1, and node v_0 ends with degree 1. The transformation also keeps the degree of $v_1, v_2 \dots v_k$ unchanged. Once this transformation has been applied to all non-leaves of T then the resulting tree T' will have only nodes of degree 1 and 3; giving a binary tree as desired.

Now observe that every original node is a leaf in T'. The only links we added were ℓ_v which have the form v'_{k+1} to v_0 where v'_{k+1} is also a leaf under the transformation. The resulting set of links L' is leaf-to-leaf.

We will now consider any feasible solution A to (T, c, L). Let $A' = A \cup \{\ell_v\}_{v \in V}$. The cost of A and A' are the same as we added only links ℓ_v which were given cost 0. First observe that ℓ_v covers all the edges of the form $v'_i v'_{i+1}$ and $v'_0 v_0$. Now consider an edge $v'_i v_i$ after the transformation. There is some link $\ell \in A$ which covers $v_0 v_i$ in T and now that same link must cover $v'_i v_i$ in T'. So, A' is a valid solution to (T', c', L') of the same cost.

Now let A' be a feasible solution (T', c', L') now consider there is a vertex v_0 which was not initially a leaf node in T. It must be the case that A' contains ℓ_v as this is the only link in L' which covers $v'_k v'_{k+1}$. So, let $A = A' - \{\ell_v\}_{v \in V}$. Now by the same argument as previously, as A' is a feasible solution for T' and the only edges in T' not in T are those covered by the ℓ_v then A is a valid solution to (T, c, L). Notice that A and A' have the same cost as we only removed links of cost 0 from the solution. □

5.3 Large Links

The main result of this section is the following theorem.

Theorem 5.3.1. Given a solution x to the EDGE-LP with $x_\ell \geq \alpha$ when $x_\ell > 0$ and m is the number of non-zero links then there exists integer solutions $x^1, x^2, \dots x^{2m}$ and $\lambda_1, \dots, \lambda_{2m}$ such that:

$$\frac{2}{1+\alpha} x \leq \sum_{i=1}^{2m} \lambda_i x^i$$

and this convex combination can be found in strongly polynomial time.

This gives an alternative proof of Cheriyan, Jordán and Ravi [13]. In particular, it gives a $\frac{4}{3}$-approximation when we start with a fractional solution where all non-zero links have weight at least $\frac{1}{2}$.

5.3.1 Algorithm

We will be working with a tree rooted at an arbitrary node, r. The least common ancestor (LCA) of a link, is the least common ancestor of its endpoints. We let L_ℓ (R_ℓ) be the path in the tree from the LCA of ℓ to the left (right) endpoint of ℓ (this path could be empty).

Given a fractional solution, x let $\alpha = \min_{\ell: x_\ell \neq 0} x_\ell$, and let $\beta = \frac{2}{1+\alpha}$. Let k be the smallest integer such that $k\beta x$ is an even integer for all entries. In order to find our convex decomposition in the algorithm below, we will decompose $k\beta x$ into k different colors such that each color is a feasible tree augmentation.

The main idea of how the algorithm works is that it goes down the tree looking at links which have their LCA at the current node and colors all the copies of each link with different colors so as to help cover the edges as much as possible with new colors. This guarantees that the first $\alpha\beta k$ links (copies of one link) which are colored through an edge all get distinct colors. Afterward, we only guarantee that of the remaining links that cover an edge half of them give a new color to that edge.

Algorithm 5 The simple coloring algorithm

Data: T a tree, x LP solution, β approximation factor, k colors
Result: Decomposition of $k\beta x$ into k different colors where each color is a feasible tree augmentation

```
Make kβx_ℓ copies of each link ℓ
while some link is not colored do
    ℓ has the highest LCA among uncolored links
    while not all copies of ℓ colored do
        Color a copy of ℓ with the first color not present on L_ℓ
        if all edges of L_ℓ are covered by all k colors then
            Color a copy of ℓ with any color not already on a copy of ℓ
        end
        Color a copy of ℓ with the first color not present on R_ℓ
        if all edges of R_ℓ are covered by all k colors then
            Color a copy of ℓ with any color not already on a copy of ℓ
        end
    end
end
```

We will now show that this coloring does indeed give us a convex combination as desired.

Theorem 5.3.2. Algorithm 5 guarantees that every edge is covered by a link in every one of the k colors.

Proof. For a given e without all k colors, every time a link through e receives a pair of colors, then one of those colors is new to e. Let us consider some link ℓ through e. Each inner while loop of the algorithm gives two colors to copies of ℓ. One of the two paths L_ℓ, R_ℓ must contain e; without loss of generality let $e \in L_\ell$. Consider the highest edge $f \in L_\ell$ without all k colors. If f is missing a color c, then e must also be missing color c. We have only colored links whose LCA is above f, therefore any link with a color which covers e must also cover f. So, for each pair of colors chosen for a link through e, at least one of them is a new color for e. In other words, half of the time a link covering e gets colored, it is a new color for e.

The first time a link through an edge e is colored, then all its colors are distinct (unless $\beta x_\ell > 1$). For a given link ℓ, every time a color is picked for a copy of ℓ it has to be a color not on one of the

edges ℓ covers or a color not on any copy of ℓ. If $\beta x_\ell > 1$, then we can color the copies of ℓ with all k colors and all the edges which ℓ covers will be covered by all k colors. In this case, e would get all k colors.

Thus the first time an edge has one of its links colored it receives at least $\alpha\beta k$ distinct colors. Combining this with the fact that every edge gets colors at rate $\frac{1}{2}$ subsequently, the total number of colors e receives in this process is at least

$$\alpha\beta k + \frac{1-\alpha}{2}\beta k = \frac{1+\alpha}{2}\beta k = k.$$

□

Now we will show how this implies Theorem 5.3.1.

Proof of Theorem 5.3.1. By scaling our x up by $k\beta$ we can write this scaled version as the sum of k different feasible colors (integer solutions). This gives us that:

$$k\beta x = \sum_{i=1}^{k} x^i$$

where the x^i are integer solutions. Dividing by k gives the desired result.

Algorithm 5 does not need to first multiply by βk before being run. The algorithm can be run by just multiplying the solution by β. As the algorithm runs, it will keep track of a convex combination of integer partial solutions. In each while loop when a link ℓ is added, ℓ will be fully added to some integer partial solutions and added to a fraction of at most two partial integer solutions (one for R_ℓ and one for L_ℓ). This creates at most two more integer partial solutions. The number of different integer solutions at the end can be bounded by $2m$ where m is the number of non-zero links. This guarantees this algorithm can be run in strongly polynomial time. □

5.3.2 Conjecture

Theorem 5.3.1 deals with the case when x does not have fractional parts which are very small. In particular, the case where $\alpha = \frac{1}{3}$ gives a $\frac{3}{2}$ approximation with this algorithm. Another approach to this problem would be to iteratively round when a solution has a link with fractional value at least $\frac{2}{3}$ (See e.g., [60]).

In particular, when a fractional solution has $x_\ell \geq \frac{2}{3}$ we can immediately round up x_ℓ to 1 and resolve the linear program with this added constraint. This approach combined with using our approximation when $x_\ell \geq \frac{1}{3}$ for all ℓ would achieve a $\frac{3}{2}$ approximation as every individual link gets rounded up by at most $\frac{3}{2}$ and the cost of the residual LPs do not increase in the process.

By combining these two approaches one would be able to provide a $\frac{3}{2}$ approximation to weighted TAP. It would be very convenient if some polynomial-time computable or every extreme point fractional solution had one of these two properties: a link ℓ with $x_\ell \geq 2/3$, or $x_\ell \geq 1/3$ for all non-zero x_ℓ. Unfortunately, there exists extreme points of the EDGE-LP which satisfy neither of these properties as shown in Section 5.5. Therefore, we propose the following conjecture.

Conjecture 5.3.3. Every extreme point solution x^* to the ODD-LP has one of the two properties: $x^*_\ell \geq 1/3$ for all non-zero x^*_ℓ or there is some ℓ such that $x^*_\ell \geq 2/3$.

5.4 Deficient Paths

In this section, we will start with a solution to NODE-LP and use the additional structure from the constraints 5.3 to help us. Due to our previous observation, we will assume that the TAP instance is a binary tree with all links going from leaf-to-leaf. We will break edges into two groups depending on how much coverage they receive.

Definition 5.4.1. An edge $e \in T$ is considered *deficient* if $x(\delta(e)) < 4/3$ and *abundant* if $x(\delta(e)) \geq 4/3$.

The deficient edges in a solution to the NODE-LP can not be too dense; this would violate the node constraint 5.3.

Lemma 5.4.2. The deficient edges form paths in T.

Proof. Suppose there was a node, v, adjacent to three deficient edges: e_1, e_2, e_3. By the node inequality 5.3, we know that:

$$x(\delta(e_1) \cup \delta(e_2) \cup \delta(e_3)) \geq 2$$

In this particular case, every link through v goes through exactly 2 of e_1, e_2, e_3. So, we have:

$$x(\delta(e_1) \cup \delta(e_2) \cup \delta(e_3)) \leq \frac{1}{2}(x(\delta(e_1)) + x(\delta(e_2)) + x(\delta(e_3))) < 2$$

This is a contradiction to the feasibility of x for the NODE-LP. So, there is no node with three deficient edges in a solution to the NODE-LP, and the deficient edges form paths as desired. □

5.4.1 A Top-Down Greedy 2-approximation and Ramifications

In this section, we present a simple 2-approximation which will be used to deal with the abundant edges in future cases. There are numerous 2-approximations for TAP, but we will use a specific coloring one as it allows us to extend colorings.

Choose any vertex r to be the root. Let k be the smallest non-negative integer such that kx_ℓ is an integer for all links ℓ. For this approach, we will multiply our fractional solution by $4k$ and then break it up into $2k$ integral solutions. The cost of the cheapest such solution will be at most $4k/2k = 2$ times the cost of the original.

We will be using LCA(ℓ), R_ℓ and L_ℓ as defined in the previous section.

The top-down algorithm is given in 6. The main idea is to double each link and use one copy to cover the left path from its LCA and the other for its right path. In this sense, it is reminiscent of the approach of Frederickson and JáJá [31] of splitting each cross link in the tree to two up links to devise a 2-approximation algorithm. The main idea of the coloring algorithm is to only supply colors to links that are missing at one of the edges it covers. Since the links are colored top down, this ensures that any color missing at an edge is also missing in all its descendant edges.

Lemma 5.4.3. The $2k$ colors returned by 6 are valid solutions.

Proof. Consider any $e \in T$. When the algorithm starts, there are at least $4k$ links which cover e, because $x(\delta(e)) \geq 1$. After the transformation of the links, there are at least $2k$ edges which cover e.

Algorithm 6 Greedily colors the links representing of the EDGE-LP top-down to give $2k$ solutions.

Input: Tree T, root r, feasible solution x to EDGE-LP, least common multiple k
Output: Breaks $4kx$ into $2k$ colors each of which is a solution
for *Links* ℓ **do**
  Break the $4kx_\ell$ into $2kx_\ell$ copies of R_ℓ and $2kx_\ell$ copies of L_ℓ
end
while *Not all* $2k$ *colors are solutions* **do**
  Let e be the highest edge without all $2k$ colors
  Choose an uncolored link ℓ in $x(\delta(e))$
  Choose a color c_i not on e
  Color ℓ with c_i
end
Use the colors L_ℓ, R_ℓ for ℓ

As the algorithm progresses, the colors covering every edge are a subset of those covering its parent. Let p be the parent of e. The first time we color a link through e that is not through p, then we must have given p all $2k$ colors already.

Every time a link through e gets a color, it is because some edge e' above e was missing that color. By the above observation, the colors missing from e' are also missing from e. Therefore, e also got a new color. Hence, every time e gets one of it's $2k$ links colored, it gets a new color.

Every edge is covered by all $2k$ colors, so every color is a solution as desired. □

The correctness of the algorithm implies that taking the cheapest color (in terms of total link cost) is a valid solution, leading to the following result.

Corollary 5.4.4. There is a greedy top-down coloring based 2-approximation for TAP.

When none of the edges are deficient, then we can push this result even further.

Corollary 5.4.5. Given a solution x to TAP with no deficient edges, $3kx$ can be decomposed into $2k$ feasible colors.

Proof. We can re-use Algorithm 6 and its proof. The only thing we have to change is that we break $3kx_\ell$ into two parts of size $\frac{3}{2}kx_\ell$. Since all edges are abundant, $x(\delta(e)) \geq 4/3$ and so after the split, every edge e has $2k$ links covering it. □

We will strengthen this further to allow us to finish off the abundant parts after we deal with the deficient parts of the tree in later proofs.

Definition 5.4.6. A rooted subtree is considered *abundant* if all its edges are abundant.

Definition 5.4.7. A partial coloring of $3kx$ causes a *conflict* if there is an edge e which is covered by three links of the same color c and e does not yet have all $2k$ colors covering it. A partial coloring is considered *conflict-free* if it causes no conflicts.

In particular, we show that given the start of the coloring we can finish it off if we didn't do too much wrong.

Theorem 5.4.8. Given a rooted partial conflict-free coloring of $3kx$ on some links with all deficient edges getting all $2k$ colors, the coloring can be extended to cover all the edges in the subtree.

Proof. We can simply start the greedy algorithm at the root and finish every edge off. In the proof of correctness of the greedy algorithm, we simply needed that the number of uncolored links was at least twice the number of colors needed. Every abundant edge has at least $4k$ links covering it originally. If the edge already has c colors, then the number of uncolored links is at most $4k - 2c$ which is twice the number of colors the edge still needs, $2k - c$. □

5.4.2 One Deficient Path

We now extend the greedy coloring algorithm to show that if the deficient edges only form one path in the tree, then there exists a $\frac{3}{2}$ approximation.

Consider we have a solution x to NODE-LP with only one deficient path P; let $u_1, u_2, \dots u_j$ be the deficient path. We will deal with this case by first coloring some links such that every edge in P gets all $2k$ colors. Then we will split up all the uncolored links that go through P. We root the tree at u_1, then we will use Theorem 5.4.8 to finish all the abundant subtrees.

Algorithm 7 Greedily colors the links to give $2k$ solutions that cover the path. Also, it avoids overcoloring the links through abundant edges and results in a conflict-free coloring.

Input: Tree T with one deficient path $P = u_1u_2 \dots u_j$, feasible solution x to NODE-LP, least common multiple k

Output: Breaks $3kx$ into $2k$ colors which cover P and is conflict-free at all abundant edges

for u_iu_{i+1} *an edge in* P **do**
  for *Color* c *not covering* u_iu_{i+1} **do**
    Pick an uncolored link, ℓ, through u_iu_{i+1} Color ℓ with color c
  end
end
while *There is some* u_iu_{i+1} *with at least three links of color* c **do**
  Let the three links through u_iu_{i+1} of color c be ℓ_1, ℓ_2, ℓ_3 With respect to the edges in P, let ℓ_1 cover only a subset of the edges covered by ℓ_2 and ℓ_3
  (At least one such labeling exists by making ℓ_2 and ℓ_3 the two links with the furthest coverage in the two sides of u_iu_{i+1} respectively)
  Uncolor ℓ_1
end

Theorem 5.4.9. Algorithm 7 provides all $2k$ colors to all the edges in P and is conflict-free.

Proof. As the algorithm progresses, an edge in P either has all $2k$ colors or all of the colored links covering it are distinct colors. If an edge e in P were to receive a duplicate color c before all $2k$ colors, then there were two edges e_1, e_2 that both needed c. Without loss of generality let e_2 lie between e and e_1. When e_1 takes color c on a link that also covers e, then the link must cover e_2.

This contradicts that e_2 would need color c. Therefore, every edge in P gets all $2k$ colors in the first part of the iteration.

The clean-up phase does not remove any colors from edges in P. A link ℓ of color c that becomes uncolored, is uncolored only if all the edges in P it covers have color c from other links. So, the clean-up phase never removes any colors from edges of P.

The clean-up phase guarantees that for all the rooted subtree, the coloring induces at most two links of that color and therefore causes no conflicts. □

Now by combining the previous theorem, and Theorem 5.4.8 there is a $\frac{3}{2}$ approximation when there is only one deficient path.

Corollary 5.4.10. Given a solution x for NODE-LP on tree T which induces at most one deficient path, there is an integral solution of cost at most $\frac{3}{2}$ the cost of x.

5.4.3 Two Deficient Paths

We've shown how to deal with a single deficient path. To extend our approach to two deficient paths, we need to deal with the abundant path of tree edges which connects the two deficient paths in the binary tree . The goal is to color links to cover the two deficient paths and the abundant path between them, and maintain that every remaining abundant subtree doesn't receive too many copies of each color. We will prove the following in this section:

Theorem 5.4.11. Given a solution x on tree T which induces at most two deficient paths, there is an integral solution of cost at most $\frac{3}{2}$ the cost of x.

To deal with the deficient edges and the abundant path between them, we will have to examine the structure of links near deficient edges.

Lemma 5.4.12. Given a solution x to the NODE-LP and a deficient edge $e = uv$ where u is an internal node, then the total weight of links through u but not e is at least $\frac{2}{3}$.

Proof. Let the neighbors of u be v, w_1, w_2. The triangle constraint on edges uv, uw_1, uw_2 says the total weight of the links that cover these edges is at least 2. All the links that cover uv, uw_1, or uw_2 also go through u. So, the total weight of links through u is at least 2. The weight of links through the deficient edge uv is less than $\frac{4}{3}$ by definition; this gives the total weight of links through u but not through uv is at least $\frac{2}{3}$ as desired. □

Consider x is a solution to NODE-LP with two deficient paths; let P_1, P_2 be the deficient paths and let $Q = q_1 q_2 \dots q_j$ be the abundant path which connects them. In order to deal with this case, we will first color the links that form the intersections of Q with P_1 and P_2. Then, we will finish coloring Q. Lastly, we will color P_1, P_2. Throughout the whole process we will also guarantee that the coloring is conflict free, i.e., every abundant edge not in Q gets all $2k$ colors or has at most two copies of every color. In order to deal with this coloring, we will have to treat links which cover all of Q differently.

Definition 5.4.13. A link ℓ is considered to be a *long link* if ℓ covers all the edges of Q. A link ℓ which is not a long link, is considered to be a *short link*.

Let e_1, e_2 be the edges of P_1 adjacent to Q. Let e_3, e_4 be the edges of P_2 adjacent to Q. If P_1 or P_2 only has one edge adjacent to Q this simplifies the case greatly. We will address this case last. Algorithm 8 finds a coloring starting at the ends of Q.

Algorithm 8 Takes two adjacent deficient edges e_1, e_2 and pairs up $2k$ of the $3kx$ links covering e_1 and e_2 in a way that avoids any edge not in Q being covered by both links in a pair unless it is covered by all pairs.

Input: Tree T, feasible solution x to NODE-LP, least common multiple k, adjacent deficient edges e_1, e_2 in P_1 between which the abundant path Q to the other deficient path P_2 originates

Output: Pairs up $2k$ links covering e_1 and $2k$ links covering e_2 in a way that avoids overusing any edge not on Q

for $i = 1$ *to* $2k$ **do**
  Let f_i be the link through e_1 which covers the ith most number of edges of Q
  Let g_i be the link through e_2 which covers the ith most number of edges of Q
end
Pair up the edges such that f_i gets paired with g_{2k+1-i}
for e *not on* Q **do**
  while f_i, g_j *are paired and both cover* e **do**
    Choose a second pair $f_{i'}, g_{j'}$ where neither covers e
    if *No such pair* $f_{i'}, g_{j'}$ *exists* **then**
      Break from the While loop
    end
    Change the pairing so that $f_i, g_{j'}$ are paired and $f_{i'}, g_j$ are paired
  end
end

Lemma 5.4.14. Algorithm 8 finds a pairing such that every edge not on Q is either covered by all $2k$ pairs or is only covered by at most one link in every pair.

Proof. Let e be an edge which had a conflict, so a swap occurred, and let e' not on Q be an edge which has more conflicts due to this swap. We will show that e' can't exist.

Consider that some swap occurred because e was covered by both f_i, g_j but not covered by $f_{i'}, g_{j'}$. If e' is covered by both links in both pairs $f_i, g_{j'}$ and $f_{i'}, g_j$ then it was covered by both links in the pairs before the swap. So, consider that e' is covered by both links $f_i, g_{j'}$ after the swap but was not covered by both links in either of the pairs before the swap. So, neither of $f_{i'}, g_j$ cover e'.

Right now we have that f_i covers e_1, e', e but not e_2; g_j covers e_2, e but not e_1, e'; $f_{i'}$ covers only e_1; $g_{j'}$ covers e_2, e' and not e_1, e.

Consider the tree T' which is all the edges of T contracted except for e, e', e_1, e_2. The edges e_1, e_2 were adjacent in T so they are still adjacent in T'. e_1 can't separate e_2 and e as g_j covers e and e_2. Similarly, e_2 can't separate e_1 and e because of f_i. Likewise, e_1 can't separate e_2 and e' because of $g_{j'}$, and e_2 can't separate e_1 and e' because of f_i. This leaves only the three possibilities shown in Figure 5.2.

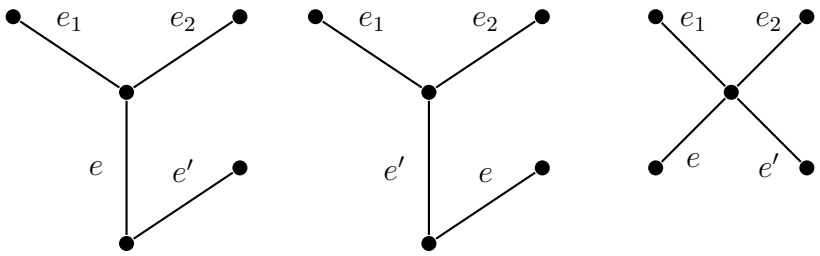


Figure 5.2: The three possible configurations of the edges e_1, e_2, e, e'

In the first case of Figure 5.2, then $g_{j'}$ must cover e as it covers e_2 and e'. This is a contradiction as $g_{j'}$ does not cover e. In the second case of Figure 5.2, g_j would cover e' as it covers e_2 and e. This is a contradiction as g_j does not cover e'. Due to f_i covering e', e_1, e, it must be the case that e_1, e, e' are all on a path; this removes the third case in Figure 5.2.

Therefore, e' doesn't exist and the swaps never increase the number of pairs which both cover an edge. When the algorithm ends, every pair covers an edge, or that edge is covered in every pair. □

Note that this swapping algorithm would work regardless of which $2k$ links through e_1, and through e_2 were chosen and how we initially paired them.

Now we can use Algorithm 8 to start a coloring on either side of the abundant path Q. We now need to coordinate the pairings and then finish coloring the abundant path. The first thing to observe, we can swap any two long links as long as the two edges of e_1, e_2, e_3, e_4 they cover are the same. This will never create any edges which have two links from the same pairing but not a link from every pairing. To coordinate the pairings on either side of Q then we need to deal with the long links. We consider them in three mutually exclusive and collectively exhaustive cases: the first case where there are at most $2k$ long links, the second when there are more than $2k$ long links, and finally, when we have $2k$ long links covering one of e_1, e_2, e_3, e_4.

At most $2k$ long links. In this case, we first observe that the initial pairing will use all the long links, and none of the long links are paired up with each other in Algorithm 8. In addition, a swap will always be initiated by a pair of short links f_i, g_j; if it were a pair of a long link and a short link, f_i, g_j would have no common edges outside of Q. So, any swap involves at most one long link. Therefore, a swap never creates a pair of two long links.

Use Algorithm 8 to pair up $2k$ links for e_1, e_2 and $2k$ links for e_3, e_4. Every long link is used in a pairing for e_1, e_2 and a pairing for e_3, e_4 and no pairing has two long links. We use Algorithm 9 to color the links to cover Q.

Lemma 5.4.15. The coloring produced by Algorithm 9 is conflict-free, when there are at most $2k$ long links.

Proof. Suppose for a contradiction that there was some edge e covered by 3 links of a color c but not covered by all $2k$ colors. This edge e is not on Q (as all edges of Q are covered by all $2k$ colors). Let q_i be the closest vertex of Q to e in T. Without loss of generality, two of the links of color c (f_1, f_2) through e also go through q_{i-1} (the other possibility is that there are two links

Algorithm 9 Takes the pairs from Algorithm 8 and extends them to cover all of Q, and be conflict free

Input: Tree T, solution x to NODE-LP, least common multiple k, adjacent deficient edges e_1, e_2, $2k$ pairs of links through e_1, e_2, adjacent deficient edges e_3, e_4, $2k$ pairs of links through e_3, e_4

Output: Colors the pairs and some other links in a conflict-free way to cover Q

for f_i, g_i *one of the* $2k$ *pairs covering* e_1, e_2 **do**
    Color f_i, g_j with an unused color c_i
    if f_i *or* g_i *is a long link* **then**
        WLOG let f_i, h_i be a pair of e_3, e_4
        Color h_i with c_i
    end
end
for $i = 1$ *to* $j - 1$ *(*j *is the number of nodes in* Q*)* **do**
    while $q_i q_{i+1}$ *doesn't have a color* c **do**
        Let f be the uncolored link covering $q_i q_{i+1}$ and the most number of edges on Q after $q_i q_{i+1}$
        Color f with color c
        if f *is in a pair for* e_3, e_4 **then**
            Let f, g be the pair
            Color g with color c
        end
    end
end

of color c through q_{i+1}. If both f_1, f_2 were part of a pair from e_1, e_2, then Algorithm 8 would guarantee e has all $2k$ colors covering it. So, without loss of generality f_1 was added to extend the coloring of color c along Q when f_2 was already colored c. If f_1 covers a subset of the edges of Q that f_2 does, then f_1 never would have been colored with color c by Algorithm 9. If f_2 covers a subset of the edges of Q that f_1 does, then f_1 would have been chosen and colored before f_2. Both of these are a contradiction to the correct running of Algorithm 9. □

This shows when there are at most $2k$ long links, we can form a conflict-free partial coloring that covers all of Q, e_1, e_2, e_3, e_4.

More than $2k$ long links In this case, we don't have to worry about covering Q as every pair will have a long link. However we will have to be a little more careful with the coordination of the two pairings. In this case, consider there are $2k + c$ long links. We will first show we can create c pairs that use two long links, and then just create the remaining $2k - c$ pairs with one long link each.

Lemma 5.4.16. If there are $2k + c$ long links, and no edge of e_1, e_2, e_3, e_4 has more than $2k$ long links, then there exists c pairs of long links such that each of these c pairs cover e_1, e_2, e_3, e_4.

Proof. Let c_{ij} be the number of links that cover e_i and e_j. A pair that covers all of e_1, e_2, e_3, e_4 consists of a long link through e_1, e_3 and a long link through e_2, e_4 or a long link through e_1, e_4 and a long link through e_2, e_3. Let $c_A = \min(c_{13}, c_{24}), c_B = \min(c_{14}, c_{23})$. If there were not c pairs that covered all of e_1, e_2, e_3, e_4 then $c_A + c_B < c$. Without loss of generality, $c_A = c_{13}, c_B = c_{14}$ so the number of long links covering e_1 is less than c. But then, the total number of long links covering e_2 is more than $2k$. This is a contradiction. Therefore, we can make c pairs such that each of these c pairs cover e_1, e_2, e_3, e_4. □

For this case, we simply create these c pairs of long links indicated in the lemma above as the first c colors. Next, we can use the remaining $2k - c$ long links as the start to fill the remaining colors using Algorithm 8. Now, if e_i isn't covered by a color, pick an uncolored link f that doesn't go through any other e_j and color f with the remaining color. This creates no conflicts. Each color consisting of two long links covers only the edges of Q twice and everywhere else once. Each color with only one long link only has two short links, and the color covers each of e_1, e_2, e_3, e_4 exactly once; each edge can only have at most two links of this color covering it.

More than $2k$ long links cover a single e_i In this case, without loss of generality e_1 has $2k$ long links. In this case, we simply start by creating and coloring the $2k$ pairs as in Algorithm 8 for e_3, e_4. Each of these pairs will cover all of Q and e_1, e_3, e_4. For each color which e_2 is missing, let f be an uncolored link covering e_2 but not e_1. Color f with the color that e_2 is missing. This again keeps the coloring conflict-free by the same reasoning as before. This gives a partial coloring that covers all of Q, e_1, e_2, e_3, e_4.

Finishing the deficient paths In all the cases above, we colored the links to create a conflict-free partial coloring that covers all of Q and e_1, e_2, e_3, e_4. To finish the deficient paths, we will use Lemma 5.4.12 to observe there are $2k$ uncolored links crossing every deficient edge. Consider starting at e_1 and moving away from Q along P_1. Let e be the deficient edge; there are $2k$ links through e not through e_1. For each color e is missing, choose one of these uncolored links for this color. Repeat this moving away from e_1 along P_1. We can do this same process along P_1 moving away from e_2. This guarantees that we only add colors to the abundant subtrees hanging off of P_1 (and not to abundant subtrees hanging off of Q or P_2). In addition, we can add each color at most twice to every subtree. We can repeat this process similarly with e_3, e_4 on P_2. This gives a conflict-free partial coloring which gives all the edges of P_1, P_2, Q all $2k$ colors. By Theorem 5.4.8, this coloring can be completed and there is a $\frac{3}{2}$ approximation when there are only two deficient paths.

End of P_1, P_2 If P_1 or P_2 have only one edge adjacent to Q the entire process above can be done simply without creating a pairing on such a side. Consider P_1 only has one edge e_1 adjacent to Q; instead of $2k$ pairs created for P_1 just take the $2k$ links through e_1 which go the furthest down Q. This replaces Algorithm 8 for P_1 and then we proceed according to the case we are in.

This proves Theorem 5.4.11 as in all the cases we have provided a $2k$ coloring of $3kx$ such that every color is a feasible solution. This coloring method could potentially be extended to the general case of multiple deficient paths, with the key difficulty being that there are links that cover segments of potentially several such deficient paths, and their colorings must be somehow globally coordinated. We have no counter examples to the success of such a potential approach even though we have no candidate algorithm that might complete the job for all feasible solutions of NODE-LP.

5.5 Comparing the linear programs

In section 5.3, we proposed a conjecture that every extreme point solution to the ODD-LP has a link which gets $x_\ell \geq 2/3$ or all the non-zero x_ℓ are at least $1/3$. This conjecture does not hold for extreme points of the EDGE-LP. An example is given in Figure 5.3.

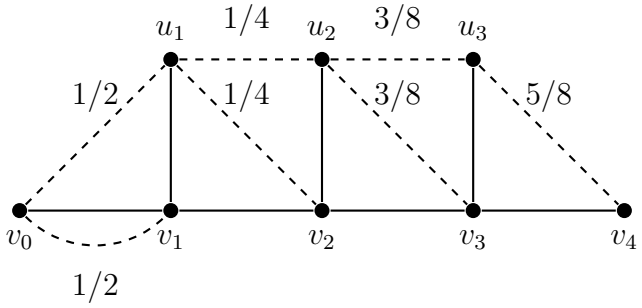
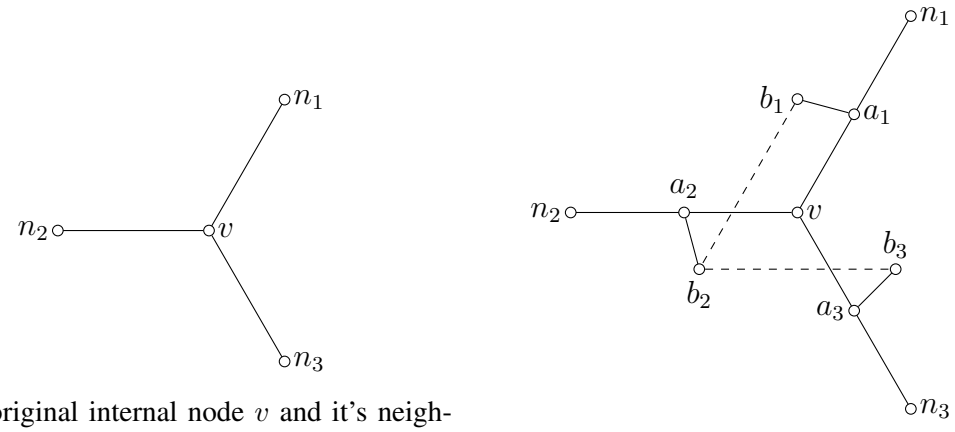


Figure 5.3: An example of an extreme point of EDGE-LP which doesn't fulfill the conjecture.



(a) The original internal node v and it's neighborhood

(b) The gadget which gets placed around v. The dotted links are added and given weight zero. No other links are added adjacent to b_1, b_2, b_3

Figure 5.4: The left figure shows the original node, and the right figure shows the node after the gadget transformation has been applied to it

We believe the conjecture to be true for extreme points of ODD-LP. We have performed exhaustive searches on all extreme points of ODD-LP to verify it on small trees (binary trees with at most 12 nodes). In addition, Fiorini et. al [28] showed that in the special case of no in-links then ODD-LP is integer. Given a rooted tree T with root r then an *in-link* is a link ℓ where ℓ does not go through r and ℓ does not go from one node to it's ancestor. This indicates that the ODD-LP potentially has more structure than the EDGE-LP that might be exploited to prove the conjecture.

In this chapter, we use the structure given from the NODE-LP (e.g. Lemma 5.4.12) to prove theorem 5.4.11. While the NODE-LP does add some constraints to the EDGE-LP, the NODE-LP is not much stronger than the EDGE-LP as we show in the next observation. In particular, we can transform any TAP instance to a slightly bigger one by a gadget expansion at every node so that any feasible solution to the EDGE-LP on the original instance is feasible to the NODE-LP in the expanded instance. This shows that if we were to do the gadget expansion for any input, the NODE-LP constraints alone (without the other ODD=LP constraints) will not add any strength to the resulting solutions.

Lemma 5.5.1. After a transformation of any TAP instance that leaves the solutions unchanged, the integrality gap for the NODE-LP on the transformed instance is the same as the integrality gap for the EDGE-LP in the original instance.

Proof. In order to show this result, we will show that for every binary tree T we can transform it to another binary tree T' such that every feasible solution of the EDGE-LP for T corresponds to a feasible solution of the NODE-LP for T'.

Consider an original node $v \in T$ with neighbors n_1, n_2, n_3 as shown in Figure 5.4a. We will transform v as shown into the structure given in Figure 5.4b.

Any feasible solution to the EDGE-LP on T can be made a feasible solution to the NODE-LP

on T' by adding all the zero-cost links with value 1 in the solution. These zero cost links guarantee all new tree edges that we added are covered. Every degree 3 node, u has at least 2 weight of links covering its neighbors.

Consider any feasible solution to the NODE-LP on T'. For every node $v$the solution must choose links b_1b_2 and b_2b_3 with value 1 for otherwise a_1b_1 and a_3b_3 will not be covered. To get a feasible solution to the EDGE-LP on T we simply remove the links b_1b_2, b_2b_3 around every node.

Neither of these transformations change the cost, so we have proven our result as desired. This gadget forces b_1, b_3 to be covered fully by the dotted links. These ensure that the node constraints around v, a_1, a_2, a_3 are all satisfied as long as the edge constraints are satisfied.

□

While the NODE-LP is not any stronger than the EDGE-LP, we believe that the ODD-LP is stronger than both of these LPs. As noted before, the ODD-LP may add key constraints which would allow us to extend the top-down deficient path coloring beyond just one or two deficient paths.

5.6 Odd Constraints

In this section, we provide a new proof of correctness of the ODD-LP. In addition, we rewrite the ODD-LP into a form which resembles the T-join LP. Lastly, we show that using this formulation we can also achieve a $\frac{3}{2}$-approximation in a special case.

Lemma 5.6.1. The constraints in ODD-LP are valid for any integer solution to TAP.

Proof by Robert Carr. Consider an odd set of vertices S. By adding together the edge constraints for $\delta(S) \cap T$ we get:

$$\sum_{e \in \delta(S) \cap T} x(\delta(e)) \geq |\delta(S) \cap T|$$

Now we can add any non-negative terms to the left hand side and still remain feasible. Therefore

$$x(\delta(S)) + \sum_{e \in \delta(S) \cap T} x(\delta(e)) \geq |\delta(S) \cap T|$$

is also feasible. Now consider any link ℓ. If x_ℓ appears an even number of times in $\sum_{e \in \delta(S) \cap T} x(\delta(e))$ then ℓ is not in $\delta(S)$. Similarly, if x_ℓ appears an odd number of times in $\sum_{e \in \delta(S) \cap T} x(\delta(e))$ then ℓ is in $\delta(S)$. So, the coefficient of every x_ℓ on the left hand side of this expression is even. In particular, for any integer solution the left hand side is even and the right hand side is odd. Therefore, we can strengthen the right hand side by increasing it by one, and the resulting constraint will still be feasible for any integer solution. The constraint

$$x(\delta(S)) + \sum_{e \in \delta(S) \cap T} x(\delta(e)) \geq |\delta(S) \cap T| + 1$$

is valid for any integer solution to TAP as desired. □

The ODD-LP is very close to being a T-join LP. Let's introduce a variable y_e for every edge e. For now we will let $\delta(S)$ denote the links with exactly one node in S, and $\delta_T(S)$ denote the edges of T with exactly one endpoint in S. If we let $y_e = x(\delta(e)) - 1$ then we can rewrite the odd constraints as:

$$x(\delta(S)) + \sum_{e \in \delta(S) \cap T} x(\delta(e)) \geq |\delta(S) \cap T| + 1$$
$$x(\delta(S)) + \sum_{e \in \delta(S) \cap T} (x(\delta(e)) - 1) \geq 1$$
$$x(\delta(S)) + \sum_{e \in \delta(S) \cap T} y_e \geq 1$$
$$x(\delta(S)) + y(\delta_T(S)) \geq 1$$

Now we can rewrite the ODD-LP as follows:

$$\min \sum_{\ell \in E} c_\ell x_\ell$$
$$x(\delta(S)) + y(\delta_T(S)) \geq 1 \qquad \forall S \subseteq V, |S| \text{ odd} \tag{5.5}$$
$$y_e = x(\delta(e)) - 1 \qquad \forall e \in T \tag{5.6}$$
$$x_\ell \geq 0 \qquad \forall \ell \in E$$

Notice that constraint 5.5 is in fact the inequalities for the up-hull of the T-join polytope [18]. This means that we could easily decompose any fractional solution (x, y) into a convex combination of T-joins easily. Unfortunately, these T-joins might contain copies of tree edges resulting from the y_e values.

The equations 5.5 and non-negtivity constraints define the dominate of the T-join polytope. Therefore we can decompose any feasible solution (x_ℓ, y_e) into T-joins. Let the decomposition be $\sum_i \lambda_i H_i$ where H_i are T-joins, and $\sum_i \lambda_i = 1$. Unfortunately, due to the y_e variables it could be the case that a tree edge is included in a T-join but that edge is not covered by any links. We call such an edge e dangerous. We let $d(e)$ denote the fraction of time that e is dangerous. By the definition of dangerous, we have $y_e \geq d(e)$.

5.6.1 Patching a tight odd cut

In this section, we will show that if we restrict our attention to one tight odd cut S that we can patch this cut. In other words, we will use the T-join decomposition and $\frac{1}{2}x$ to guarantee the edges in the cut are all covered by some link.

Lemma 5.6.2. Given a feasible solution (x, y) to the modified ODD-LP and a tight odd cut S, there exists a decomposition of $\frac{3}{2}x$ into a convex combination of integer solutions, such that every integer solution covers $\delta_T(S)$.

Proof. Let us start with a decomposition of (x, y) into T-joins $\sum_i \lambda_i H_i$. We will give a matching of links to edges such that every edge e gets covered to the extent at least $d(e)$.

S is crossed by exactly weight 1 on edges and links. Therefore, each T-join H_i can have at most one edge of $\delta(S)$ be dangerous (as every T-join crosses S exactly once).

$\frac{x_f}{2}$ is the fraction of time link f can be used to cover. Consider the bipartite graph $G = (L, R, E)$ where L is the set of links, R is the set of edges $\delta_T(S)$ and there is an edge $uv \in E$ iff $u \in L, v \in R$ and the link corresponding to u covers the edge corresponding to v.

In order to find a patching, we will find a fractional matching in G such that every vertex in $u \in R$ is used at least $d(u)$ and every vertex in $v \in L$ is used at most $\frac{x_v}{2}$. To show this we will check Hall's condition; in particular we will show that for any set $U \subseteq R$ we have $d(U) \leq x(N(U))/2$.

Consider that U contains three edges which are incomparable, then by the feasibility of the solution, the weight of links covering these edges is at least 2. We know the total dangerous of $d(R)$ is at most 1 as every T-join misses at most 1 edge in R. So, we get that $1 \leq 2/2$ as desired.

Now consider that U has no three edges which are incomparable, that means that U is a set of edges $u_1, u_2, ...u_k$ which all lie on a path in T. Without loss of generality let $u_1, \ldots u_k$ be the edges in order on this path. By constraint 5.6 the coverage of u_1 must be at least $1 + d(u_1)$. The $d(u_i)$ of the time that u_i is dangerous, there is a link covering u_{i+1}, as u_{i+1} is not dangerous. These $d(u_i)$ links must be disjoint from the links covering u_i otherwise u_i would not be dangerous. Therefore, the total coverage of these edges is at least

$$1 + d(u_1) + d(u_1) + d(u_2) + \cdots + d(u_{k-1}).$$

Without loss of generality, $d(u_1) \geq d(u_k)$. Therefore, we get the desired result, our total coverage is

$$1 + d(u_1) + d(u_1) + d(u_2) + \cdots + d(u_{k-1}) \geq 1 + d(u_1) + \cdots + d(u_k) \geq 2(d(u_1) + \cdots + d(u_k))$$

Hence, we can find a matching. This allows us to use $\frac{1}{2}x$ to patch up the T-joins we got from decomposing (x, y) and cover all the edges of $\delta_T(S)$.

□

Unfortunately, this approach does not give a patching directly, nor does it give a clear way to extend to even two tight cuts.

5.7 Three-Cycle TAP

5.7.1 Weighted Version

In this section, we will consider the weighted version of 3TAP where the weights on the links, c_f, can take on any value. We first present an $O(\log n)$ approximation algorithm, and then we present a matching lower bound of $\Omega(\log n)$, where n is the number of nodes in the tree.

Theorem 5.7.1. There is a $O(\log n)$ approximation algorithm for weighted 3TAP on n nodes.

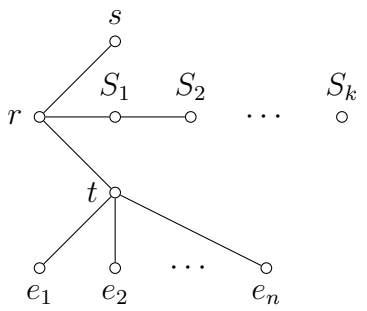


Figure 5.5: The 3TAP instance created from a set cover instance.

Proof. Consider any feasible solution A to 3TAP, such that $T \cup A$ has every tree edge in a 3-cycle. For a vertex v, let $\delta(v)$ be the edges of $T \cup A$ adjacent to v.

To turn this problem into a set cover problem, we let the edges of $E(T)$ be the elements. For any subset of edges adjacent to a vertex v, S_v, we construct a set with cost $c(S_v)$ which covers the edges induced by the endpoints of the edges in S_v (except for those edges adjacent to v).

By doubling each edge in the feasible solution, then we can decompose the entire solution into these stars. So, given a solution to 3TAP of total cost C, the corresponding set cover has a solution of at most $2C$. Given any solution to the set cover problem, then we simply add all the edges specified by the stars to the tree (with maybe some duplicates when edges are added from both endpoints' stars). Thus, any solution of cost C to the set cover gives a solution to the original 3TAP solution of cost between $C/2$ and C. Therefore the optimal solutions to these two problems are within a factor of two of each other.

It is well known that minimum-cost set cover with n elements has an $O(\log n)$ approximation as long as the densest set (that has the maximum ratio of newly covered elements divided by the cost of the set) can be found in polynomial time. For a fixed vertex v, we can find the maximum density star centered at v as follows. Due to a result by Goldberg, one can find the maximum density subgraph; $S \subset V$ which minimizes $\frac{|E(S)|}{c(S)}$ in polynomial time [39]. For the given center v, by setting the cost of another vertex u to be $c(uv)$, we can use the maximum density subgraph algorithm to find the maximum density star from v, where the edges in the subgraph are the tree edges covered in triangles by the corresponding star. By repeating this for every choice of center vertex v, we can find the maximum density star in polynomial time. This gives the maximum density set for the set cover problem in polynomial time. Then we can use the greedy algorithm for set cover to get an $O(\log n)$ approximation for 3TAP. □

Notice that in the above algorithm we used no properties of the original graph T. This algorithm will in fact work for any graph T where the goal is to augment such that every edge is in a 3-cycle.

Corollary 5.7.2. The problem of finding a minimum cost augmentation of any graph G where every edge must be in a 3-cycle has an $O(\log n)$ approximation.

The above approximation is tight as the weighted 3TAP problem captures set-cover exactly. We will now show the matching lower bound.

Theorem 5.7.3. 3TAP does not have a $\Omega(\log n)$-approximation unless $NP \subseteq P$.

Proof. Consider an instance of set cover with sets $S_1, S_2, \ldots, S_k$ and elements $e_1, e_2, \ldots e_n$ and cost function c on the sets. We will have our tree be as shown in Figure 5.5. The vertex set is

$$\{r, s, t\} \cup \{S_i\}_{i=1}^{k} \cup \{e_j\}_{j=1}^{n}$$

with the following costs on the links:

- Links from s to vertices $\{r, t\} \cup \{S_i\}_{i=1}^{k}$
- Links from t to S_i have cost $c(S_i)$
- If $e_j \in S_i$ then the link from e_j to S_i has cost 0
- All remaining links have cost $1 + \sum_{i=1}^{k} c(S_i)$. Call the set of these remaining edges L.

In any optimal solution, we will not use any edges from L as taking all the edges not in that set have smaller cost and give a feasible solution. The zero edges from s allow every edge except for the te_j edges to be in a three-cycle and they have cost 0. Now the only way to have an edge te_j in a three cycle is for tS_i and e_jS_i to be used for some S_i such that $e_j \in S_i$. So, the non-zero edges bought correspond to sets being chosen.

Consider any feasible solution to set cover $S_{i_1}, \ldots S_{i_k}$, this can be turned into a feasible solution to 3TAP of the same cost. All the zero cost edges in addition to the tS_{i_ℓ} edges form a feasible solution. We know all the edges in the tree except for the te_j edges are in a three cycle with zero cost edges. Consider any $j \in [n]$. There is some S_{i_t} that contains j. The edge te_j is then in a three cycle with tS_{i_ℓ} and $S_{i_\ell}e_j$. Hence, every feasible solution to the set cover instance gives a feasible solution of the same cost to the 3TAP instance.

Consider any feasible solution to our 3TAP instance. If the 3TAP solution contains an edge from L then the solution has weight at least $1 + \sum_{i=1}^{k} c(S_i)$, by taking all the sets S_i we get a feasible solution to the set cover instance of less cost. Now consider there are no edges from L in the feasible solution for the 3TAP instance. Let $tS_{i_1}, \ldots tS_{i_t}$ be the non-zero edges in the solution. Therefore $S_{i_1}, \ldots S_{i_t}$ is a feasible solution to the set cover instance. Consider any element e_j. The edge te_j must be in some three cycle with tS_{i_ℓ} and S_{i_ℓ} therefore, S_{i_ℓ} contains e_j and is a set in our solution to set cover. Therefore every feasible solution to 3TAP has a corresponding solution of set cover with the same or smaller cost.

Any feasible solution to set cover gives a solution to 3TAP of the same cost. Any feasible solution to 3TAP, gives a feasible solution to set cover of the same or smaller cost. Therefore, by the hardness of set cover, it is impossible to approximate three-cycle TAP to within a $\Omega(\log n)$ factor unless $NP \subseteq P$ [4]. □

Remark: Suppose we were given an empty initial graph to augment and wish to find a minimum-cost two-edge-connected spanning subgraph where every edge is in a triangle, it is not hard to adapt the above hardness: We give all edges in the tree zero cost. By further subdividing the path of set nodes $S_1, S_2, \ldots, S_k$ to add new dummy nodes between every pair of set nodes, we can ensure that every element node e_j is covered only by triangles containing edge (t, e_j). This requires that the other edges in the cycle are of the form $(t, S_i), (S_i, e_j)$ for some set S_i containing the element e_j.

5.7.2 Unweighted Version

While weighted 3TAP has many similarities to set cover, the unweighted version admits a constant approximation unlike set cover. Here we consider the case that every non-tree edge has cost either 1 or infinity, and every tree edge is present (and has cost 0). This 4-approximation comes from lower bounding the cost of every feasible solution to unweighted 3TAP.

Lemma 5.7.4. Every feasible unweighted 3TAP solution has cost at least $\frac{n-1}{2}$.

Proof. Consider any solution S. Duplicate all the links of S and edges T and forming stars around every vertex consisting of the edges adjacent to it. Call the star around v, S_v. This doubles the cost of the solution, but now we can see that every tree edge is covered by some star. At every vertex, we can further decompose S_v into $S_v^1, \ldots S_v^{\ell_v}$ such that we get stars that cover different connected components of the tree and every star contains at most one tree edge.

Now consider any star S_v^i. If S_v^i has x links, then the number of tree edges it can cover with 3-cycles is at most x. So, in the doubled instance of S there must be at least $n-1$ edges. Every link is in at most 2 stars; there must be at least $\frac{n-1}{2}$ edges in any feasible solution. □

Corollary 5.7.5. Unweighted 3TAP has a 4-approximation.

Proof. We can get a 4 approximation by simply taking any minimal feasible solution. For every edge ab, pick a v such that av, bv both have cost 0 or 1. If no such vertex exists, then no feasible solution exists. Otherwise, the algorithm chooses at most $2(n-1)$ links. This gives a 4 approximation as desired. □

5.8 Conclusion

We have introduced a new top down coloring method that gives a strict improvement over existing 2-approximation algorithms for weighted TAP, with better improvements for larger minimum values in the LP. Our methods give constructive convex combinations into feasible solutions and when coupled with the strengthened ODD-LP for the problem have much potential to settle the integrality gap for this fundamental network design problem. We also settled the approximation complexity of the special case when all edges in the final solution must be in triangles – the extensions to short constant-length cycles in place of triangles is immediate. We hope our new algorithms will provide a stepping stone to settling the integrality gap for weighted TAP.

Chapter 6

Conclusion and Open Questions

In chapter 2, we have obtained new results in the approximability of rumor spreading problems in the well-studied radio model as well as a new model motivated by wireless communications, which we call the edge-star model. For the radio model, we present an $\Omega(n^{1/2-\epsilon})$ hardness of approximation bound for radio gossip, making progress on an open problem mentioned in [36]. For the edge-star model, we present an $O(\frac{\log n}{\log \log n})$ approximation algorithm for gossip, an $\tilde{O}(2^{\sqrt{\log n}})$ approximation algorithm for symmetric multicommodity multicast, and an $\tilde{O}(n^{2/3})$ approximation algorithm for asymmetric multicommodity multicast. Our approximation algorithms expose relationships between the edge-star model and the well-studied telephone model.

Our work leaves several interesting open problems. Among the nine cells listed in the matrix of Table 2.1 of Section 2.1, only radio broadcast and edge-star broadcast are resolved. Significant gaps between the best known upper and lower bounds on approximability remain for telephone broadcast, the gossip problem under all three models, and the multicommodity multicast problem under all three models. In the edge-star model, the symmetric and asymmetric versions of the multicommodity multicast problem are distinct, and both are open, in terms of the best approximation factor achievable in polynomial-time.

In chapter 3, we continued the study of these communication problems in planar graphs. We gave the first proof of an upper bound of $O(\log k)$ on the integrality gap of the POISE-LP. We utilized the poise result combined with path separators to develop an $O(\log^3 k \frac{\log n}{\log \log n})$-approximation for telephone multicommodity problem on planar graphs. In addition, we develop an $O(\log^3 n)$-approximation for radio gossip on planar graphs. Lastly, we extend these results to graphs which are minor-free. One natural open problem is if these path separator techniques can extend to the other communication problems in planar graphs. In particular, does the path separator technique or poise problem allow for better approximations for the edge-star model in planar graphs.

In chapter 4, we studied publisher-subscriber problems. We gave constant factor approximations in the case where the demands were complete bipartite. In the case of the complete demand graph, then we showed that an optimal solution is a hub-star and can be found in polynomial time. For general demands, we addressed the tree-star case and gave an $O(\log n)$-approximation algorithm. Lastly, we gave an $\Omega(\log \log n)$ integrality gap on the natural LP relaxation.

Perhaps the most interesting set of questions arising from our work is the approximation status

of the bipartite and more general DON problems with general demands. While the integrality gap demonstrates that an LP rounding approach based on the natural formulation will not yield constant guarantees, we have not yet been able to convert these ideas into a hardness of constant-factor approximation proof.

Another avenue of open problems concern removing the metric requirement on the cost function: the tree-DON and tree-star-DON problems then become significantly different and harder (they generalize group Steiner problem, for instance) and require new ideas.

Finally, there exists $O(\log n)$-approximations for all three design metrics, but the techniques used are specific to each problem. One further avenue would be to find a unified $O(\log n)$-approximation for all three design types:star-star, tree-tree, tree-star.

In our last chapter, we introduced new top-down coloring methods that gives a strict improvement over existing 2-approximation algorithms for weighted TAP, with better improvements for larger minimum values in the LP. Our methods give constructive convex combinations into feasible solutions and when coupled with the strengthened ODD-LP for the problem have much potential to settle the integrality gap for this fundamental network design problem. We also settled the approximation complexity of the special case when all edges in the final solution must be in triangles – the extensions to short constant-length cycles in place of triangles is immediate. We hope our new algorithms will provide a stepping stone to settling the integrality gap for weighted TAP. In particular, resolving whether conjecture 5.3.3 is true or not would provide insight into the ODD-LP. Another open problem is just to simply achieve a better than 2-approximation for general weighted TAP.

Bibliography

[1] Ittai Abraham and Cyril Gavoille. Object location using path separators. In *PODC*, pages 188–197, 2006.

[2] David Adjiashvili. Improved approximation for weighted tree augmentation with bounded costs. *arXiv preprint arXiv:1607.03791*, 2016.

[3] N. Alon, A. Bar-Noy, N. Linial, and D. Peleg. A lower bound for radio broadcast. *J. Comput. Syst. Sci.*, 43:290–298, 1991.

[4] Sanjeev Arora and Madhu Sudan. Improved low-degree testing and its applications. *Combinatorica*, 23(3):365–426, 2003.

[5] B. Awerbuch and D. Peleg. Sparse partitions. In *Foundations of Computer Science, 1990. Proceedings., 31st Annual Symposium on*, pages 503–513. IEEE, 1990.

[6] Brenda Baker and Robert Shostak. Gossips and telephones. *Discrete Mathematics*, 2(3):191–193, 1972.

[7] A. Bar-Noy, S. Guha, J. Naor, and B. Schieber. Multicasting in heterogeneous networks. In *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing, Dallas, Texas, USA, May 23-26, 1998*, pages 448–453, 1998.

[8] Amotz Bar-Noy, Sudipto Guha, Joseph Naor, and Baruch Schieber. Message multicasting in heterogeneous networks. *SIAM J. Comput.*, 30(2):347–358, 2000.

[9] Robert Carr and Santosh Vempala. Randomized metarounding. In *Proceedings of the thirty-second annual ACM symposium on Theory of computing*, pages 58–62. ACM, 2000.

[10] R. C. Chakinala, A. Kumarasubramanian, K. A. Laing, R. Manokaran, C. Pandu Rangan, and R. Rajaraman. Playing push vs pull: models and algorithms for disseminating dynamic data in networks. In *SPAA*, pages 244–253, 2006.

[11] Joseph Cheriyan and Zhihan Gao. Approximating (unweighted) tree augmentation via lift-and-project, part I: stemless TAP. *CoRR*, abs/1508.07504, 2015.

[12] Joseph Cheriyan and Zhihan Gao. Approximating (unweighted) tree augmentation via lift-and-project, part ii. *arXiv preprint arXiv:1507.01309*, 2015.

[13] Joseph Cheriyan, Tibor Jordán, and R Ravi. On 2-coverings and 2-packings of laminar families. *Algorithms-ESA99*, pages 72–72, 1999.

[14] Joseph Cheriyan, Howard Karloff, Rohit Khandekar, and Jochen Könemann. On the integrality ratio for tree augmentation. *Operations Research Letters*, 36(4):399–401, 2008.

[15] B.V. Cherkassky. Mnogopolyusnye dvukhproduktovye zadachi [russian: Multiterminal two commodity problems]. *Issledovaniya po Diskretnoi Optimizatsii [Russian: Studies in discrete optimization]*, pages 261–289, 1976.

[16] Nachshon Cohen and Zeev Nutov. A (1+ ln2)-approximation algorithm for minimum-cost 2-edge-connectivity augmentation of trees with constant radius. *Theoretical Computer Science*, 489:67–74, 2013.

[17] Yevgeniy Dodis and Sanjeev Khanna. Designing networks with bounded pairwise distance. In *STOC*, pages 750–759, 1999.

[18] Jack Edmonds and Ellis L Johnson. Matching, euler tours and the chinese postman. *Mathematical programming*, 5(1):88–124, 1973.

[19] Friedrich Eisenbrand, Fabrizio Grandoni, Thomas Rothvoss, and Guido Schafer. Connected facility location via random facility sampling and core detouring. *JCSS*, 76(8):709 – 726, 2010.

[20] M. Elkin and G. Kortsarz. Polylogarithmic additive inapproximability of the radio broadcast problem. *SIAM J. Discrete Math*, 19(4):881–899, 2005.

[21] M. Elkin and G. Kortsarz. An improved algorithm for radio broadcast. *ACM Transactions on Algorithms (TALG)*, 3(1):8, 2007.

[22] Michael Elkin and Guy Kortsarz. A combinatorial logarithmic approximation algorithm for the directed telephone broadcast problem. *SIAM J. Comput.*, 35(3):672–689, 2005.

[23] Michael Elkin and Guy Kortsarz. An approximation algorithm for the directed telephone multicast problem. *Algorithmica*, 45(4):569–583, 2006.

[24] Michael Elkin and Guy Kortsarz. Sublogarithmic approximation for telephone multicast. *J. Comput. Syst. Sci.*, 72(4):648–659, 2006.

[25] Jittat Fakcharoenphol, Satish Rao, and Kunal Talwar. A tight bound on approximating arbitrary metrics by tree metrics. *STOC*, pages 448–455, 2003.

[26] U. Feige and J. Kilian. Zero knowledge and the chromatic number. *J. Comput. Syst. Sci.*, 57:187–199, 1998.

[27] Uriel Feige, David Peleg, Prabhakar Raghavan, and Eli Upfal. Randomized broadcast in networks. *Random Structures & Algorithms*, pages 447–460, 1990.

[28] Samuel Fiorini, Martin Groß, Jochen Könemann, and Laura Sanità. A 3/2-approximation algorithm for tree augmentation via chvátal-gomory cuts. *arXiv preprint arXiv:1702.05567*, 2017.

[29] Nikolaos Fountoulakis, Konstantinos Panagiotou, and Thomas Sauerwald. Ultra-fast rumor spreading in social networks. In *Proceedings of the Twenty-third Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '12, pages 1642–1660. SIAM, 2012.

[30] A. Frank. *Connections in Combinatorial Optimization*. Oxford Lecture Series in Mathematics and Its Applications. OUP Oxford, 2011.

[31] Greg N Frederickson and Joseph Jájá. Approximation algorithms for several graph augmen-

tation problems. *SIAM Journal on Computing*, 10(2):270–283, 1981.

[32] Greg N Fredrickson and Joseph Jájá. On the relationship between the biconnectivity augmentation and traveling salesman problem. *Theoretical Computer Science*, 19(2):189–201, 1982.

[33] Rajiv Gandhi, Magnús M Halldórsson, Christian Konrad, Guy Kortsarz, and Hoon Oh. Radio aggregation scheduling. In *Proceedings of the 11th International Symposium on Algorithms and Experiements for Wireless Sensor Networks (ALGOSENSORS)*, 2015.

[34] M. R. Garey and D. S. Johnson. *Computers and intractability.* Freeman Press, 1979.

[35] Naveen Garg, Goran Konjevod, and R. Ravi. A polylogarithmic approximation algorithm for the group Steiner tree problem. *J. Algorithms*, 37(1):66–84, 2000.

[36] L. Gasieniec. On efficient gossiping in radio networks. In Shay Kutten and Janez erovnik, editors, *Structural Information and Communication Complexity*, volume 5869 of *Lecture Notes in Computer Science*, pages 2–14. Springer Berlin Heidelberg, 2010.

[37] L. Gasieniec, D. Peleg, and Q. Xin. Faster communication in known topology radio networks. *Distributed Computing*, 19(4):289–300, 2007.

[38] George Giakkoupis. Tight bounds for rumor spreading in graphs of a given conductance. In Thomas Schwentick and Christoph Dürr, editors, *28th International Symposium on Theoretical Aspects of Computer Science (STACS 2011)*, volume 9 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 57–68, Dagstuhl, Germany, 2011. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[39] Andrew V Goldberg. *Finding a maximum density subgraph*. University of California Berkeley, CA, 1984.

[40] Daniel Golovin, Viswanath Nagarajan, and Mohit Singh. Approximating the k-multicut problem. In *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, pages 621–630. Society for Industrial and Applied Mathematics, 2006.

[41] Navin Goyal, Neil Olver, and F. Bruce Shepherd. The VPN conjecture is true. *J. ACM*, 60(3):17, 2013.

[42] Michelangelo Grigni and David Peleg. Tight bounds on minimum broadcast networks. *SIAM J. Discrete Math.*, 4(2):207–222, 1991.

[43] Anupam Gupta, Jon M. Kleinberg, Amit Kumar, Rajeev Rastogi, and Bülent Yener. Provisioning a virtual private network: a network design problem for multicommodity flow. *STOC*, pages 389–398, 2001.

[44] András Hajnal, Eric C Milner, and Endre Szemerédi. A cure for the telephone disease. *Canad. Math. Bull*, 15(3):447–450, 1972.

[45] Eran Halperin, Guy Kortsarz, Robert Krauthgamer, Aravind Srinivasan, and Nan Wang. Integrality ratio for group Steiner trees and directed Steiner trees. *SODA*, pages 275–284, 2003.

[46] Jennifer Iglesias, Rajmohan Rajaraman, R Ravi, and Ravi Sundaram. Designing overlapping networks for publish-subscribe systems. *Approximation, Randomization, and Combinatorial*

Optimization. Algorithms and Techniques, page 381, 2015.

[47] Jennifer Iglesias, Rajmohan Rajaraman, R. Ravi, and Ravi Sundaram. Rumors across radio, wireless, telephone. In *FSTTCS*, pages 517–528, 2015.

[48] Jennifer Iglesias, Rajmohan Rajaraman, R Ravi, and Ravi Sundaram. Plane gossip: Approximating rumor spread in planar graphs. *arXiv preprint arXiv:1612.01492*, 2017.

[49] Jennifer Iglesias and R Ravi. Coloring down: $3/2$-approximation for special cases of the weighted tree augmentation problem. *arXiv preprint arXiv:1707.05240*, 2017.

[50] Kamal Jain. A factor 2 approximation algorithm for the generalized steiner network problem. *Combinatorica*, 21(1):39–60, 2001.

[51] R. Karp, C. Schindelhauer, S. Shenker, and B. Vocking. Randomized rumor spreading. In *Proceedings of the 41st Annual Symposium on Foundations of Computer Science*, FOCS '00, pages 565–, Washington, DC, USA, 2000. IEEE Computer Society.

[52] R. M. Karp, F. T. Leighton, R. L. Rivest, C. D. Thompson, U. V. Vazirani, and V. V. Vazirani. Global wire routing in two-dimensional arrays. *Algorithmica*, 2(1):113–129, Nov 1987.

[53] Richard M Karp. Reducibility among combinatorial problems. In *Complexity of computer computations*, pages 85–103. Springer, 1972.

[54] Samir Khuller and Ramakrishna Thurimella. Approximation algorithms for graph augmentation. *Journal of Algorithms*, 14(2):214–225, 1993.

[55] Guy Kortsarz and Zeev Nutov. A 1.75 lp approximation for the tree augmentation problem. *arXiv preprint arXiv:1507.03009*, 2015.

[56] Guy Kortsarz and Zeev Nutov. A simplified 3/2 ratio approximation algorithm for the tree augmentation problem. *Transaction on Algorithm*, 12(2):23, 2016.

[57] Guy Kortsarz and David Peleg. Approximation algorithms for minimum-time broadcast. *SIAM J. Discrete Math.*, 8(3):401–427, 1995.

[58] Guy Kortsarz and David Peleg. Generating low-degree 2-spanners. *SIAM Journal on Computing*, 27(5):1438–1456, 1998.

[59] D. R. Kowalski and A. Pelc. Optimal deterministic broadcasting in known topology radio networks. *Distributed Computing*, 19(3):185–195, 2007.

[60] Lap Chi Lau, Ramamoorthi Ravi, and Mohit Singh. *Iterative methods in combinatorial optimization*, volume 46. Cambridge University Press, 2011.

[61] Shi Li. Approximation algorithms for network routing and facility location problems. *Doctoral Dissertation, Princeton University*, Januray 2014.

[62] Richard J Lipton and Robert Endre Tarjan. A separator theorem for planar graphs. *SIAM J. Applied Mathematics*, 36(2):177–189, 1979.

[63] L. Lovász. On some connectivity properties of eulerian graphs. *Acta Mathematica Academiae Scientarum Hungaricae*, 28:129–138, 1976.

[64] Fredrik Manne, Shuang Wang, and Qin Xin. Faster radio broadcast in planar graphs. In

WONS, pages 9–13, 2008.

[65] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge International Series on Parallel Computation. Cambridge University Press, 1995.

[66] A. Nikzad and R Ravi. Sending secrets swiftly: Approximation algorithms for generalized multicast problems. In *Automata, Languages, and Programming*, pages 568–607. Springer, 2014.

[67] Zeev Nutov. A note on the tree augmentation problem. *arXiv preprint arXiv:1703.07247*, 2017.

[68] Laura J. Poplawski and Rajmohan Rajaraman. Multicommodity facility location under group steiner access cost. *SODA*, pages 996–1013, 2011.

[69] Andrzej Proskurowski. Minimum broadcast trees. *IEEE Trans. Computers*, 30(5):363–366, 1981.

[70] R Ravi. Rapid rumor ramification: Approximating the minimum broadcast time. In *Foundations of Computer Science, 1994 Proceedings., 35th Annual Symposium on*, pages 202–213. IEEE, 1994.

[71] R. Ravi. *Steiner Trees and Beyond: Approximation Algorithms for Network Design*. PhD thesis, Brown University, 1994.

[72] R Ravi. Matching based augmentations for approximating connectivity problems. In *LATIN*, pages 13–24. Springer, 2006.

[73] R. Ravi and F. Sibel Salman. Approximation algorithms for the traveling purchaser problem and its variants in network design. *ESA*, pages 29–40, 1999.

[74] R. Ravi and A. Sinha. Multicommodity facility location. *SODA*, pages 342–349, 2004.

[75] Thomas Rothvoss and Laura Sanita. On the complexity of the asymmetric VPN problem. In *APPROX/RANDOM*, pages 326–338, 2009.

[76] David B. Shmoys, Chaitanya Swamy, and Retsef Levi. Facility location with service installation costs. In *SODA*, pages 1088–1097, 2004.

[77] Chaitanya Swamy and Amit Kumar. Primal-dual algorithms for connected facility location problems. *Algorithmica*, 40(4):245–269, 2004.

[78] Chaitanya Swamy and David B. Shmoys. Fault-tolerant facility location. In *SODA*, pages 735–736, 2003.

[79] Mikkel Thorup. Compact oracles for reachability and approximate distances in planar digraphs. *J. ACM*, 51(6):993–1024, 2004.

[80] R Tijdeman. On a telephone problem. *Nieuw Archief voor Wiskunde*, 3(19):188–192, 1971.

[81] David P. Williamson, Michel X. Goemans, Milena Mihail, and Vijay V. Vazirani. A primal-dual approximation algorithm for generalized Steiner network problems. In *STOC*, pages 708–717, 1993.

[82] David P Williamson and David B Shmoys. *The Design of Approximation Algorithms*. Cam-

bridge University Press, 2011.