Minimum communication cost spanning trees and metric embeddings

Muhammad Samir Khan

Master of Science

School of Computer Science

McGill University

Montreal, Quebec

August 13, 2015

A thesis submitted to McGill University in partial fulfillment of the requirements of the degree of Master of Science

ACKNOWLEDGEMENTS

First and foremost, I am grateful to God, the most gracious and merciful, for enabling me to pursue graduate studies at McGill. I am also thankful to my parents and siblings for their unconditional love and support.

I would like to express my deepest appreciation to both my supervisors, Professor Bruce Shepherd and Professor Adrian Vetta, for their advice and guidance. I am grateful to Professor Bruce Shepherd for all the time and effort he devoted to help me with the thesis. He was always there to guide me and was always patient when results were not forthcoming.

In these past two years, I was fortunate to have many friends who have made this journey even more memorable. I would like to thank Ahmad, Saad, Shafqat, Jad, Joseph, Omar, Matthias, Liana, Richard, and many others for their wonderful company and friendship.

ABSTRACT

Given an undirected weighted graph $G = (V, E, w)$ and a demand D_{uv} for each pair of nodes $u, v \in V$, the minimum communication cost spanning tree (MCST) problem is to find a spanning tree T of G that minimizes $\sum_{u,v \in V} D_{uv} \cdot d_T(u, v)$, where $d_T(u, v)$ is the distance between the nodes u and v in T. MCST was introduced by T. C. Hu [19] and is known to be NP-hard [20]. It is also related to two problems in the well trodden field of metric embeddings. One is to find a distribution over trees with low expected stretch of any edge, and the other is to find a single tree with low average stretch of the edges of a multigraph. In this thesis, we survey the results in metric embeddings and describe their relationship to MCST. We also present Linear Programming based formulations in an attempt to improve the existing approximation guarantees.

ABRÉGÉ

Étant donné un graphe non orienté et pondéré $G = (V, E, w)$ et une demande D_{uv} pour chaque paire de sommets $u, v \in V$, le problème de l'arbre couvrant de poids minimal de communication (en anglais, minimum communication spanning tree (MCST) problem) consiste à trouver un arbre couvrant T de G qui minimise $\sum_{u,v \in V} D_{uv} \cdot d_T(u, v)$, où $d_T(u, v)$ est la distance entre les sommets u et v de T. Le problème MCST a été présenté par TC Hu [19] et est connu pour être NP-difficile [20]. Il est également lié à deux problèmes dans le domaine bien développé de plongements métriques. Le premier consiste à trouver une distribution sur les arbres aux arcs ayant un faible étirement espéré. Le second consiste à trouver un seul arbre avec un faible étirement moyen des arcs d'un multi-graphe. Dans cette thèse, on présente les résultats de plongements métriques de la littérature et on décrit leur relation à MCST. Également, on présente des formulations basées sur la programmation linéaire dans le but d'améliorer les garanties d'approximation déjà établies.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1
Introduction

1.1 Introduction

Graph sparsification has been an intensely active research area for two decades now (c.f. the survey in [8]). The general paradigm is, given a graph G, find a simpler graph H that performs approximately as well for the task at hand. "Simpler" usually means sparser, i.e. with fewer edges. The canonical sparse graph is a tree (since it is the minimal connected graph). The "task at hand" varies depending on the application, e.g. cut sparsifiers [24] and spectral sparsifiers [8] have had some sensational recent success. Our main interest is in sparsifiers H that do well in terms of communication cost for a set of demands. For an undirected weighted graph $G = (V, E, w)$, with the induced shortest path metric $d_G(\cdot, \cdot)$ on V and a demand D_{uv} for each pair of nodes $u, v \in V$, the communication cost of G is given by $\sum_{u,v \in V} D_{uv} \cdot d_G(u, v)$. Formally, the cornerstone problem in this thesis is the following.

MINIMUM COMMUNICATION COST SPARSIFIER

Input: A weighted graph $G = (V, E, w)$ with demands D_{uv} on the pairs of nodes $u, v \in V(G)$.

Output: A graph that obeys some sparsification requirements and, subject to this, minimizes the communication cost.

We are especially interested in the following special case which is a classical problem in Combinatorial Optimization.

MINIMUM COMMUNICATION COST SPANNING TREE (MCST)

Input: A weighted graph $G = (V, E, w)$ with demands D_{uv} on the pairs of nodes $u, v \in V(G)$.

Output: A spanning tree of G that minimizes the communication cost.

This problem was introduced by T. C. Hu [19] and he gave an especially elegant answer in the case where G is complete and $w(e) = 1$ for all e (i.e. the graph is unweighted). This is based on the notion of a Gomory-Hu tree, defined in Section 1.3.

Theorem 1.1.1. *The minimum communication cost spanning tree for a complete unweighted graph $G = (V, E)$ with demands D_{uv} for each $u, v \in V(G)$ is the Gomory-Hu Tree with edge capacities D_{uv} for each edge $uv \in E(G)$.*

Hu [19] also gave sufficient conditions for the minimum communication cost spanning tree to be a star. However, for general weights the problem is known to be NP-hard [20], even in restricted settings such as complete weighted graphs with uniform demands [28]. So, people have tended to study approximation algorithms for MCST. Much of this thesis is dedicated to approximation of MCST and related problems. One such related problem is to minimize the average stretch of the edges of a multigraph. For a spanning tree T of a graph G, the stretch of an edge $e = uv \in E(G)$ is given by $d_T(u, v)/d_G(u, v)$.

MINIMUM AVERAGE STRETCH SPANNING TREE (MAST)

Input: A weighted multigraph $G = (V, E, w)$ with edge multiplicities m_{uv} for every

edge $uv \in E(G)$.

Output: A spanning tree of G that minimizes the average stretch.

There are two types of guarantees that people look for when considering low communication cost trees. One is to guarantee, or bound, the cost against the best tree (i.e. approximation algorithms for MCST) and the other is to guarantee against the communication cost of the original graph itself. The latter problem is related to finding low average stretch trees and probabilistic embeddings. We study these in Chapters 2 and 3. We present their connection to MCST in Chapter 4. In Chapter 4, we also propose new linear programming formulations and discuss our attempts to bound their integrality gaps.

1.2 Preliminaries

We denote a weighted graph by $G = (V, E, w)$ where $V(G)$ and $E(G)$ represent its node set and edge set respectively, and $w : E \to \mathbb{R}^+$ are the positive *weights* (also called *lengths*) on the edges. If we have two graphs, say G and H, we use the notation w_G and w_H to distinguish between the two. For a subgraph H of G, the edge weight function in H is restricted to the edges of H. Graphs may be undirected or directed. An edge e can be represented by its endpoints as $e = uv$ for the undirected (simple) case and as $e = (u, v)$ for the directed (simple) case. The definition of weight can be extended to sets of edges (and to graphs) as well, e.g. a path P has length $w(P) = \sum_{e \in E(P)} w(e)$. For an *unweighted* or *unit-weight* graph G, we have $w(e) = 1$ for each edge $e \in E(G)$. When working with unit-weight graphs we may drop the reference to w. We think of $w(e)$ as a cost or distance associated with the edge e. For any pair of nodes $u, v \in V(G)$, let $d_G(u, v)$ be the length of a shortest path from

u to v, i.e.

$$d_G(u,v) = \min w(P) : P \text{ is a } uv\text{-path.}$$

Note that $d_G(\cdot,\cdot)$ induces a (semi-)metric space on $V(G)$. For a graph $G = (V,E)$ and a set of nodes $S \subset V$, $G[S]$ is the subgraph of G induced by S. More precisely we have $E[S] = \{e = uv \mid u \in S,\ v \in S\}$ and $G[S] = (S, E[S])$. In the case of a multigraph G, each edge $uv \in E(G)$ has edge multiplicity $m_{uv} > 0$. We use $M = \sum_{uv \in E(G)} m_{uv}$.

For an undirected graph $G = (V,E)$, let $\delta_G(u)$ denote the set of edges which have u as one of its endpoints. For the directed case, let $\delta^+_G(u)$ be the set of outgoing edges and $\delta^-_G(u)$ be the set of incoming edges. For a set $S \subset V(G)$, we denote by $\delta_G(S)$ the set of edges with exactly one endpoint in S. The directed case is defined similarly as $\delta^+_G(S) = \{e = (u,v) \mid u \in S, v \notin S\}$ and $\delta^-_G(S) = \{e = (u,v) \mid u \notin S, v \in S\}$. We drop the subscript G when it is clear from the context. A set of edges of the form $\delta(S)$, for some $S \subset V$, is called a *cut*. For two nodes, s and t, a cut is an *st-cut* if $s \in S$ and $t \in V \setminus S$. Let the *capacities* on edges be assigned by the function $c : E \to \mathbb{R}^+$. Then the size of an st-cut is given by $c(\delta(S)) = \sum_{e \in \delta(S)} c(e)$. A *min st-cut* is an st-cut with the minimum size.

For a simple graph $G = (V,E,w)$, we may *identify* $u \in V$ with $v \in V$. This creates a new node w which replaces u and v. The neighbours of w in G are the union of the neighbours of u and v in G.

A tree may be rooted at some node called the *root*. This naturally creates directions in the tree via the induced *ancestor* and *descendant* relations between

nodes. In this case, a subtree rooted at v consists of the tree on all the descendants of v with v as the root.

We call a graph *biconnected* if the graph remains connected after the removal of any node. The maximal biconnected subgraphs are called its biconnected components.

1.3 Gomory-Hu Trees

Consider a graph $G = (V(G), E(G))$ with edge capacities $c(e)$, for all $e \in E(G)$, and a tree $T = (V(T) = V(G), E(T))$. For an edge $e = st \in E(T)$, let S_e and T_e be the two connected components of $T - e$ such that $s \in S_e$ and $t \in T_e$. For two nodes $s, t \in V(T)$, let P_{st} be the unique st-path in T and let $e_{st} = \text{argmin}_{e \in P_{st}} c(\delta_G(S_e))$. Then T is called a *Gomory-Hu tree* of G if, for any $s, t \in V(G)$, $\delta_G(S_{e_{st}})$ is a min st-cut in G (Gomory and Hu [16]).

1.4 Some Special Graphs

1.4.1 Outerplanar Graphs

A graph is called *planar* if it can be drawn in a plane (i.e. $\mathbb{R}^2$) in such a way that no edges intersect each other except at their endpoints. This drawing is called its *planar embedding*. In a planar embedding, regions formed by removing images of the nodes and edges from $\mathbb{R}^2$ are called *faces*. The infinite face is called the *unbounded face*.

An *outerplanar* graph is a planar graph such that all the nodes lie on the unbounded face of its planar embedding. There exists a natural *ear decomposition* for any outerplanar graph G. We denote this by $G_0, G_1, \ldots, G_\ell = G$ where G_0 is either a path or a cycle and G_i is obtained from G_{i-1} by attaching a path P_i to either a

single node on the outerface of G_{i-1} or to the endpoints of an edge e_i on the outerface of G_{i-1}. The internal nodes (if any) of P_i are distinct from G_{i-1} and P_i is called an *ear*. Note that if G is biconnected, then we can choose G_0 as a cycle and P_i to be connected to the endpoints of an edge on the outerface of G_{i-1}. We sometimes denote the ear decomposition by $\langle P_0, P_1, \ldots, P_\ell \rangle$.

We also refer to outerplanar graphs as 1-outerplanar. For $k > 1$, a graph is called *k-outerplanar* if removing all the nodes (and their incident edges) on the unbounded face of its planar embedding gives a $(k-1)$-outerplanar graph.

1.4.2 Series-Parallel Graphs

Series-parallel graphs are defined recursively using two operations: the *Series* and *Parallel* Operations. Each series-parallel graph also has two special nodes called the *terminals*. The base case is a graph on a single edge st where s and t are the two terminals. Let $G_1 = (V(G_1), E(G_1))$ and $G_2 = (V(G_2), E(G_2))$ be two disjoint series-parallel graphs with terminals $s_1, t_1 \in V(G_1)$ and $s_2, t_2 \in V(G_2)$ respectively. The two operations are defined as follows.

Series Operation: Take G to be the union of G_1 and G_2, and then identify s_2 with t_1. The terminals of G are $s = s_1$ and $t = t_2$.

Parallel Operation: Take G to be the union of G_1 and G_2, and then identify s_1 with s_2, to get s, and identify t_1 with t_2, to get t. The terminals of G are s and t.

A canonical example of a series-parallel graph is the *diamond graph*. This is a family of series parallel graphs, $\{G_k\}_{k \in \mathbb{N}}$, which are defined inductively. G_0 is a single edge $s_0 t_0$. The k-th diamond graph $G_k = (V_k = V(G_k), E_k = E(G_k))$ has terminals

$s_k, t_k \in V_k$ and is defined as follows. Let H_k be the series composition of two disjoint copies of G_{k-1}. Then G_k is the parallel composition of two copies of H_k. See Figure 1–1 for some examples of the diamond graph. Note that we have

$$|V_k| = n_k = 4n_{k-1} - 4 \implies n_k = \frac{4^{k+1} + 8}{6} \tag{1.1}$$

and

$$|E_k| = m_k = 4m_{k-1} \implies m_k = 4^k. \tag{1.2}$$

Note also that

$$m_k = \Theta(n_k) \tag{1.3}$$

and

$$k = \Theta(\log n_k). \tag{1.4}$$

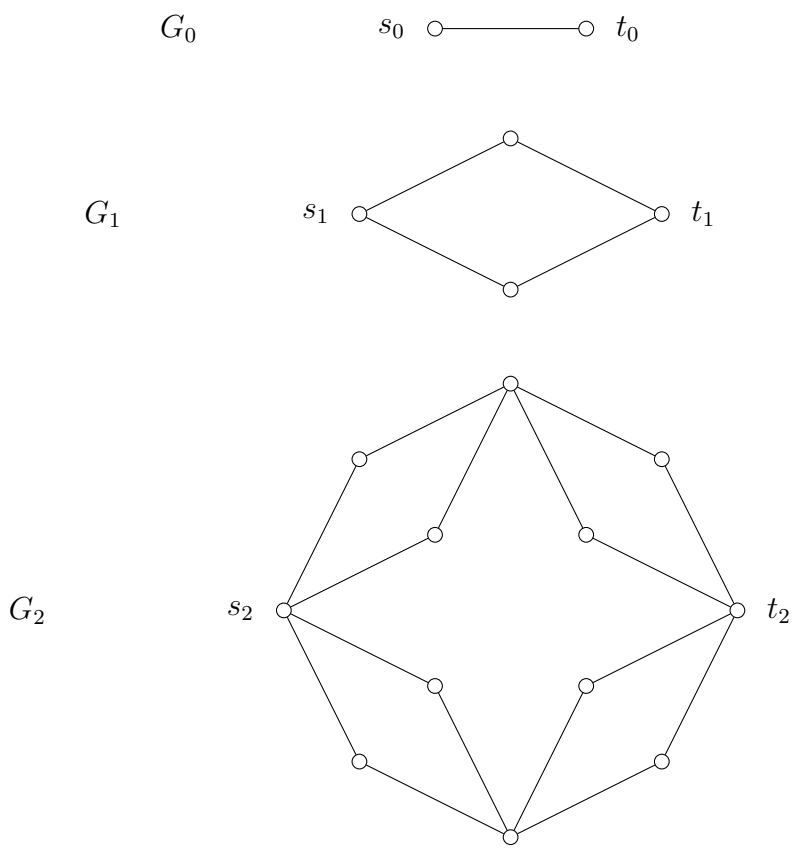


Figure 1–1: The diamond graphs G_0, G_1, and G_2.

CHAPTER 2
Embedding Graphs into Trees

2.1 Introduction to Tree Embeddings

We are interested in replacing a graph by a simpler one while maintaining its essential routing properties. In this chapter, we focus on maintaining the shortest path values. We consider a weighted graph $G = (V(G), E(G), w)$. Recall from Chapter 1 that $d_G(u, v)$ denotes the weight of a shortest uv-path in G. The following definition plays a central role.

Definition 2.1.1. *A subgraph H of G has distortion α if for any two nodes $u, v \in V(G)$ we have*

$$d_H(u, v) \leq \alpha \cdot d_G(u, v).$$

One initial observation is that to find low distortion subgraphs of G it is sufficient to bound the distortion over the edges of G.

Lemma 2.1.2. *A subgraph H of G has distortion α if and only if, for every edge $e = xy \in E(G)$ such that e is a shortest xy-path in G, $d_H(x, y) \leq \alpha \cdot d_G(x, y)$.*

Proof: The forward direction is trivially satisfied by the definition of distortion. For the reverse direction, consider any two nodes $u, v \in V(G)$. Let P_{uv} be a shortest uv-path in G. Then, for any subgraph H,

$$d_H(u, v) \leq \sum_{e=xy \in P_{uv}} d_H(x, y) \leq \sum_{e=xy \in P_{uv}} \alpha \cdot d_G(x, y) = \alpha \cdot d_G(u, v),$$

where the second inequality follows from the fact that any edge $e = xy \in P_{uv}$ is a shortest xy-path in G. $\square$

The *stretch*, in subgraph H, of any pair of nodes u, v, denoted by $\text{str}_H(u, v)$, is defined as:

$$\text{str}_H(u, v) = \frac{d_H(u, v)}{d_G(u, v)}.$$

If α is the distortion of H, then $\alpha \geq \text{str}_H(u, v) \geq 1$. The stretch of an edge $e = uv$ is given by the stretch of its endpoints, i.e. $\text{str}_H(e) = \text{str}_H(u, v)$.

Trees are the ideal routing subgraphs with respect to sparsity (but perhaps not with respect to distortion). It is well known, however, that spanning trees suffer from distortion $\Omega(n)$ even for a simple unit-weight cycle. Removal of any edge in a cycle creates a tree, but the removed edge will have stretch $n - 1$ giving the lower bound. This example motivates the use of probability distributions over a set of spanning trees instead of just a single tree. For a random tree T from a set of trees $\mathcal{T}$ under a probability distribution $\mathcal{D}$, we use the notation $T \in_{\mathcal{D}} \mathcal{T}$.

Definition 2.1.3. *A probability distribution $\mathcal{D}$ over a set of spanning trees $\mathcal{T}$ α-approximates G if for every $u, v \in V(G)$*

$$\mathbb{E}_{T \in_{\mathcal{D}} \mathcal{T}}[d_T(u, v)] \leq \alpha \cdot d_G(u, v).$$

We say that G is α-probabilistically embedded (or probabilistically embedded with distortion α) into $\mathcal{T}$ (under $\mathcal{D}$).

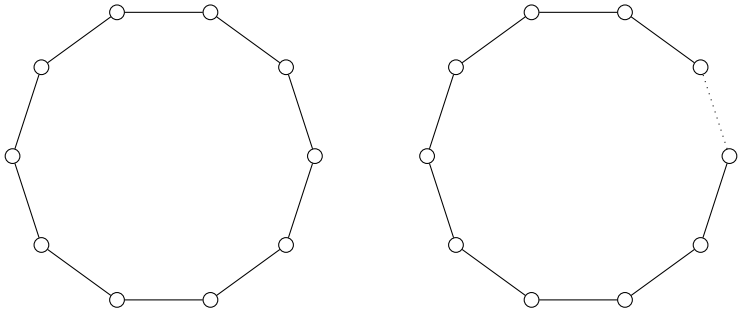


Figure 2–1: Obtaining a tree from a cycle.

Note that, as before, it is sufficient to consider only the edges of G since, using P_{uv} to denote a shortest uv-path in G, we have

$$\mathbb{E}[d_T(u,v)] \leq \mathbb{E}[\sum_{e=xy\in P_{uv}} d_T(x,y)] = \sum_{e=xy\in P_{uv}} \mathbb{E}[d_T(x,y)],$$

where the only equality follows from the linearity of expectation. Thus, if $\mathbb{E}[d_T(x,y)] \leq \alpha \cdot d_G(x,y)$ for every edge $e = xy \in E(G)$, then $\mathbb{E}[d_T(u,v)] \leq \alpha \cdot d_G(u,v)$ for every $u, v \in V(G)$ and so G can be α-probabilistically embedded into $\mathcal{T}$.

Returning to the unit-weight cycle, consider the uniform probability distribution on which edge to remove. That is, of the n possible trees we take each tree with probability $\frac{1}{n}$. The expected stretch of any edge e is then

$$\mathbb{E}[\mathrm{str}_T(e)] = 1 \cdot \frac{n-1}{n} + (n-1) \cdot \frac{1}{n} = 2 \cdot \frac{n-1}{n} \leq 2.$$

So, the unit-weight cycle can be probabilistically embedded into its spanning trees with constant distortion. In fact, this is true for any weighted cycle [21].

Lemma 2.1.4 (Karp [21])**.** *Any n-node weighted cycle can be probabilistically embedded into its spanning trees with distortion* 2*.*

Proof: Let C_n be the n-node weighted cycle. We define the probability distribution $\mathcal{D}$ over the spanning trees as follows. Note that removal of any edge from C_n gives a spanning tree. We remove one edge e from C_n at random, with probability $p(e) = w(e)/w(C_n)$, to get the tree T. For any edge $e = uv$, if it is removed from C_n, then $d_T(u, v)$ is $(w(C_n) - w(e))$ and it is $w(e)$ otherwise. These events happen with probability $p(e)$ and $1 - p(e)$ respectively. Hence, for an edge $e = uv$, we have

$$\begin{aligned}
\mathbb{E}(d_T(u,v)) &= p(e) \cdot (w(C_n) - w(e)) + (1 - p(e)) \cdot w(e) \\
&= \frac{w(e)}{w(C_n)} \cdot (w(C_n) - w(e)) + \frac{w(C_n) - w(e)}{w(C_n)} \cdot w(e) \\
&= 2 \cdot \frac{w(C_n) - w(e)}{w(C_n)} \cdot w(e) \\
&\leq 2 d_G(u,v).
\end{aligned}$$

This completes the proof. □

Graphs may also be embedded into trees that are not necessarily spanning (and thus may have different edge weights from G as well). This leads to the definition of *dominating trees* as follows.

Definition 2.1.5. *A tree T (not necessarily spanning) is called a dominating tree of G if*

(1) $V(T) \supseteq V(G)$

(2) $d_T(u, v) \geq d_G(u, v)$ *for every* $u, v \in V(G)$.

The set of nodes $V(T) \setminus V(G)$ are called Steiner Nodes.

Note that a spanning tree is also a dominating tree (but not vice versa) and thus any (probabilistic) embedding into spanning trees gives a (probabilistic) embedding

into dominating trees as well (but not vice versa). Stretch and distortion naturally extend to dominating trees. Similarly, it is again sufficient to minimize the stretch only over the edges of G. However, as with spanning trees, dominating trees also suffer from distoration $\Omega(n)$ even on simple graphs. The example used is again the unit-weight cycle, but the proof is more involved. The following theorem is due to Rabinovich and Raz [23].

Theorem 2.1.6 (Rabinovich and Raz [23])**.** *Any dominating tree of a unit-weight cycle has distortion at least* $n/3 - 1$*.*

Rabinovich and Raz gave topological arguments in [23], but a simpler, purely combinatorial proof was offered by Anupam Gupta [17] of an $\Omega(n)$ lower bound. We present this in Section 2.5 as Theorem 2.5.1. This again motivates the use of probability distributions over a set of dominating trees.

Definition 2.1.7. *A probability distribution* $\mathcal{D}$ *over a set of dominating trees* $\mathcal{T}$ *α-approximates* G *if for every* $u, v \in V(G)$

$$\mathbb{E}_{T \in_{\mathcal{D}} \mathcal{T}}[d_T(u, v)] \leq \alpha \cdot d_G(u, v).$$

We say that G *is α-probabilistically embedded (or probabilistically embedded with distortion α) into* $\mathcal{T}$ *(under* $\mathcal{D}$*).*

2.2 Dominating Trees: $\Omega(\log n)$ Lower Bound

We show a lower bound of $\Omega(\log n)$ for distortion of probabilistic embeddings of series parallel graphs into dominating trees (and hence spanning trees) due to Gupta et al. [18]. This was not the first logarithmic lower bound result. Bartal [6] gave

a $\Omega(\log n)$ lower bound for expander graphs. It shows, however, that even simple graphs may suffer $\Omega(\log n)$ distortion.

Theorem 2.2.1 (Gupta et al. [18])**.** *For the infinite family of unit-weight diamond graphs $\{G_k\}_{k\in\mathbb{N}^+}$ any α-probabilistic embedding into dominating trees has $\alpha = \Omega(\log n_k)$.*

We refer the reader to the defnition of the diamond graph (Section 1.4). We recall that $|E_k| = m_k = 4^k$ and $k = \Theta(\log(n_k))$ by Equations (1.2) and (1.4).

Proof: As before it is sufficient to consider only the maximum stretch on edges of G_k. It is also sufficient to show that for any dominating tree T

$$\sum_{e=uv\in E_k} d_T(u,v) = \Omega(k)\cdot \sum_{e=uv\in E_k} d_{G_k}(u,v) = \Omega(k)\cdot 4^k.$$

Since then any distribution over dominating trees will have distortion $\Omega(k) = \Omega(\log n_k)$ (using Equation (1.4)) as follows.

$$\mathbb{E}[\sum_{e=uv\in E_k} d_T(u,v)] = \Omega(k)\cdot 4^k = \Omega(k)\cdot |E_k|.$$

Using linearity of expectation, we get

$$\sum_{e=uv\in E_k} \mathbb{E}[d_T(u,v)] = \Omega(k)\cdot |E_k|$$

and thus

$$\frac{1}{|E_k|}\sum_{e=uv\in E_k} \mathbb{E}[d_T(u,v)] = \Omega(k).$$

So, there exists at least one edge xy such that

$$\mathbb{E}[\mathrm{str}_T(x,y)] = \mathbb{E}[d_T(x,y)] \geq \frac{1}{|E_k|}\sum_{e=uv\in E_k} \mathbb{E}[d_T(u,v)] = \Omega(k),$$

where the first equality is because we are considering the unit-weight graph.

Let T be a dominating tree of G_k. Let $S_i \subseteq E_k$ be the set of all edges that have stretch at least $2^{i+1}/3 - 1$, i.e. $S_i = \{e \mid \text{str}_T(e) \geq 2^{i+1}/3 - 1\}$, for $i \geq 1$. By the construction of G_k, any simple cycle containing s_k and t_k has length 2^{k+1} and thus, by Theorem 2.1.6, any such cycle will have at least one edge in S_k. Consider removing the edges S_k. This must separate the terminals of at least one of the four copies of G_{k-1} in G_k (otherwise we have a cycle containing s_k and t_k). Thus, $|S_k|$ is at least the size of the min $s_{k-1}t_{k-1}$-cut in G_{k-1}. By the construction of G_k, the size of a minimum $s_k t_k$-cut is 2^k and so we have $|S_k| \geq 2^{k-1}$. Using a similar argument on the four copies of G_{k-1}, S_{k-1} will get a contribution of at least 2^{k-2} edges from each copy and thus $|S_{k-1}| \geq 4 \cdot 2^{k-2}$. Arguing in the same manner we get $|S_i| \geq 4^{k-i} \cdot 2^{i-1} = 2^{2k-1-i}$.

Now, for each edge $e \in E_k$, we have

$$\begin{aligned}
\max_{i:e\in S_i}\Big(\frac{2^{i+1}}{3} - 1\Big) &= \max_{i:e\in S_i}\Big(\frac{2^{i+1}-1}{3} - \frac{2}{3}\Big) \\
&= \Big(\sum_{i=0}^{\max(i:e\in S_i)} \frac{2^i}{3}\Big) - \frac{2}{3} \\
&\geq \Big(\sum_{i=1}^{\max(i:e\in S_i)} \frac{2^i}{3}\Big) - \frac{2}{3} \\
&= \Big(\sum_{i:e\in S_i} \frac{2^i}{3}\Big) - \frac{2}{3} \\
&= \frac{1}{2}\Big(\big(\sum_{i:e\in S_i} \frac{2^{i+1}}{3}\big) - \frac{1}{3}\Big) \\
&\geq \frac{1}{2}\sum_{i:e\in S_i}\Big(\frac{2^{i+1}}{3} - 1\Big)
\end{aligned}$$

and thus

$$\begin{aligned}
\sum_{e=uv\in E_k} d_T(u,v) &\geq \sum_e \max_{i:e\in S_i} \Big(\frac{2^{i+1}}{3} - 1\Big) \\
&\geq \frac{1}{2}\sum_e \sum_{i:e\in S_i} \Big(\frac{2^{i+1}}{3} - 1\Big) \\
&= \frac{1}{2}\sum_{i=1}^{k} |S_i| \cdot \Big(\frac{2^{i+1}}{3} - 1\Big) \\
&\geq \frac{1}{2}\sum_{i=1}^{k} 2^{2k-i-1} \cdot \Big(\frac{2^{i+1}}{3} - 1\Big) \\
&= \frac{4^k}{2}\sum_{i=1}^{k} \frac{1}{2^{i+1}} \cdot \Big(\frac{2^{i+1}}{3} - 1\Big) \\
&= 4^k \cdot \sum_{i=1}^{k} \Big(\frac{1}{6} - \frac{1}{2^{i+2}}\Big) \\
&\geq \Big(\frac{k}{6} - \frac{1}{4}\Big)4^k \\
&= \Omega(k) \cdot 4^k,
\end{aligned}$$

where the last inequality follows from the fact that $\sum_{i=1}^{k} \frac{1}{2^i} \leq 1$. □

2.3 Dominating Trees: $O(\log n)$ Upper Bound

We now turn to upper bounding distortion in general graphs. Bartal [6] was the first to formally consider probabilistic embeddings (although it was implicit in [3]) and he gave an $O(\log^2 n)$ (later improved to $O(\log n \log\log n)$ in [7]) upper bound on the distortion using, what he called, *hierarchically well-separated trees*. These are so named because the weights between successive levels of the trees change by a constant factor (this was important for several of his applications). Fakcharoenphol et al. [14] gave a polynomial time algorithm to probabilistically embed any graph into

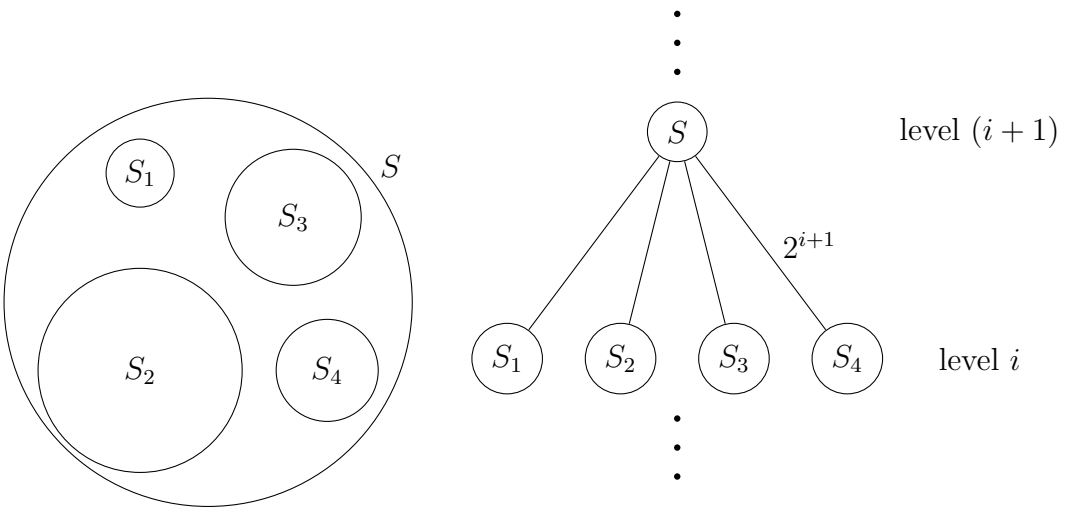


Figure 2–2: Converting a laminar family into a rooted tree.

a distribution over dominating trees with distortion $O(\log n)$. This is tight because of the $\Omega(\log n)$ lower bound (see Section 2.2). We present Fakcharoenphol et al.'s technique here.

For a graph G, a *laminar family* $\mathcal{F} \subseteq 2^{V(G)}$ is a family of subsets of $V(G)$ such that for any $A, B \in \mathcal{F}$, either $A \subseteq B$ or $B \subseteq A$ or $A \cap B = \emptyset$. Let $L(T)$ denote the leaves of a tree T. For a rooted tree T we associate a laminar family $\mathcal{F}$. For each node $v \in T$, let C_v denote the set (called a *cluster*) of leaves that are descendants of v. In particular, there is a singleton cluster for each leaf and $L(T)$ is the cluster C_r where r is the root. Clearly this is reversible. So, to each laminar family we can associate a rooted tree.

For a laminar family $\mathcal{F}$, associated with a rooted tree T, we say that a cluster $S \in \mathcal{F}$ is *ρ-centered* at a node v (not necessarily in S) if, for each $u \in S$, we have that $d_G(u, v) \leq \rho$. We may also say that S has radius ρ (this means that it is ρ-centered

at some node). Note that this implies a diameter of at most 2ρ for the cluster. Let Δ be the diameter of G and let $\gamma = \lceil \log_2 \Delta \rceil$. Each node in T is at some *level* in the tree. The root is at level γ. The level decreases with increasing distance from the root. We say that a cluster $S \in \mathcal{F}$ is at level $i+1$ if its associated node v_S in T is at level $i+1$. Fakcharoenphol et al. ensure that a cluster at level $i+1$ has radius at most 2^{i+1}. We assume WLOG that the edge weights are strictly greater than 1. This ensures that all clusters at level 0 are singleton clusters and so this is the lowest level in T.

In order to ensure that T is a dominating tree, the weight of each edge in T is assigned as follows. For a set $S \in \mathcal{F}$ at level $i+1$, let v_S be its associated node in T and let $S_1, \ldots, S_k$ be its maximal subsets. So, the clusters $S_1, \ldots, S_k$ are at level i. Assign each edge $v_S v_{S_j} \in E(T)$, for $1 \leq j \leq k$, a weight of 2^{i+1} (recall that the radius of S is at most 2^{i+1} and hence its diameter is at most 2^{i+2}). If two nodes u and v are in the same cluster S at level $i+1$ but are separated into two different clusters at level i, then $d_T(u,v) \geq 2 \times 2^{i+1} = 2^{i+2} \geq d_G(u,v)$, where the last inequality is because S has diameter at most 2^{i+2}. Thus, such a tree is a dominating tree.

The algorithm to create such clusters is now as follows. Let $V(G) = \{v_1, \ldots, v_n\}$. Select a random permutation π of $v_1, \ldots, v_n$ which is used throughout the algorithm. Initially, $V(G)$ is set to be at level γ. For each level $i+1$, we create child clusters

at level i as follows. First select a β_i from $[2^{i-1}, 2^i]$ randomly with a uniform distribution[1] (probability density function $p(x) = 1/2^{i-1}$). Consider a cluster S at level $i+1$. Assign $u \in S$ to the first (according to π) node $v \in V$ within distance β_i of u. Each child cluster of S is then the set of nodes in S that are assigned to the same center v. Note that the cluster consisting of nodes within distance β_i from v may not contain v itself even if $v \in S$. Clearly this creates a partition of S. The radius of each cluster is at most $\beta_i \leq 2^i$ and hence its diameter is at most 2^{i+1}, as desired for a cluster at level i.

Theorem 2.3.1 (Fakcharoenphol et al. [14])**.** *Any weighted graph G can be α-probabilistically embedded into a distribution over dominating trees with $\alpha = O(\log n)$.*

Proof: We analyze the algorithm described above. For an input graph G the algorithm outputs a tree T associated with a laminar family $\mathcal{F} \subseteq 2^{V(G)}$. We show that indeed, for any edge $uv \in E(G)$, we have $\mathbb{E}[d_T(u,v)] \leq O(\log n) \cdot d_G(u,v)$. Consider one step of the algorithm, where we have clusters at level $i+1$ and new ones are being created at level i. We say that a center x *settles* an edge $uv \in E(G)$ at level i if u and v are both in the same cluster at level $i+1$ and x is the first (according to π) node to which exactly one of u and v is assigned at level i (and so u and v are in different clusters at level i). We blame $d_T(u,v)$ to x using $d_T^x(u,v) = \sum_i \mathbb{1}(x \text{ settles } uv \text{ at level } i) \cdot 2^{i+3}$, where $\mathbb{1}(\cdot)$ denotes the indicator

[1]This is a different distribution than the Fakcharoenphol et al. paper [14] but not surprising, since Kunal Talwar also mentions in his PhD thesis [27] that the uniform distribution is sufficient.

function. Note that only one center may settle any edge and so $\sum_x d_T^x(u,v) = 2^{i+3}$. Recall that an edge from a level $i+1$ node in T to a level i node in T is assigned a weight of 2^{i+1}. So, for an edge $uv \in E(G)$, if u and v are both in the same cluster at level $i+1$ but are separated into two different clusters at level i, then $d_T(u,v) \leq \sum_{j=0}^{i} 2 \times 2^{j+1} \leq 2^{i+3} = \sum_x d_T^x(u,v)$.

Consider an edge $uv \in E(G)$ and an ordering of the nodes, $x_1, \ldots, x_n$, in increasing distance from uv (the distance of x from an edge uv is $\min\big(d_G(x,u), d_G(x,v)\big)$). Now, consider a node x_j and assume WLOG that $d_G(x_j,u) \leq d_G(x_j,v)$. In order for x_j to settle uv at level i, three events must occur. We label them by A_j^i, B_j^i, and C^i as follows.

- A_j^i is the event that $d_G(x_j,u) \leq \beta_i < d_G(x_j,v)$. This ensures that only u can be assigned to x_j.
- B_j^i is the event that x_j is the first (according to π) center within distance β_i of uv. This ensures that at least one of u and v is assigned to x_j.
- C^i is the event that both u and v are in the same cluster at level $i+1$. This ensures that no center has yet settled uv.

We thus have

$$\mathbb{P}[x_j \text{ settles } uv \text{ at level } i] = \mathbb{P}[A_j^i \wedge B_j^i \wedge C^i] \leq \mathbb{P}[A_j^i \wedge B_j^i] = \mathbb{P}[B_j^i | A_j^i] \cdot \mathbb{P}[A_j^i].$$

Let $a(i,j) = \max(d_G(x_j,u), 2^{i-1})$ and let $b(i,j) = \min(d_G(x_j,v), 2^i)$. Then

$$\mathbb{P}[A_j^i] = \begin{cases} \int_{a(i,j)}^{b(i,j)} \frac{1}{2^{i-1}} dx & \text{if } b(i,j) \geq a(i,j) \\ 0 & \text{otherwise.} \end{cases}$$

Now, conditioned on A_j^i, let $x_1, \ldots, x_j, \ldots, x_k$ be the centers within distance β_i of uv. Then $B_j^i | A_j^i$ is the event that x_j is the first among $x_1, \ldots, x_k$ in the permutation π. Thus

$$\mathbb{P}[B_j^i | A_j^i] = \frac{1}{k} \leq \frac{1}{j}.$$

We can now bound the expected value of $d_T^{x_j}(u, v)$ as follows.

$$\begin{aligned}
\mathbb{E}[d_T^{x_j}(u,v)] &= \sum_i \mathbb{P}[A_j^i \wedge B_j^i \wedge C^i] \cdot 2^{i+3} \\
&\leq \sum_i \mathbb{P}[A_j^i \wedge B_j^i] \cdot 2^{i+3} \\
&= \sum_i 2^{i+3} \cdot \mathbb{P}[B_j^i | A_j^i] \cdot \mathbb{P}[A_j^i] \\
&\leq \sum_{i: a(i,j) \leq b(i,j)} 2^{i+3} \cdot \frac{1}{j} \cdot \int_{a(i,j)}^{b(i,j)} \frac{1}{2^{i-1}} \cdot dx \\
&= \sum_{i: a(i,j) \leq b(i,j)} \frac{16}{j} \cdot \int_{a(i,j)}^{b(i,j)} 1 \cdot dx \\
&= \frac{16}{j} \cdot \int_{d_G(x_j,u)}^{d_G(x_j,v)} 1 \cdot dx \\
&= \frac{16}{j} \cdot (d_G(x_j, v) - d_G(x_j, u)) \\
&\leq \frac{16}{j} \cdot d_G(u, v),
\end{aligned}$$

where the last inequality follows from the triangle inequality (recall that $d_G(\cdot, \cdot)$ is a (semi-)metric).

Now, by using linearity of expectation, we can bound the expected value of $d_T(u,v)$ as follows.

$$\mathbb{E}[d_T(u,v)] \leq \sum_{j=1}^{n} \mathbb{E}[d_T^{x_j}(u,v)] \leq \sum_{j=1}^{n} \frac{16}{j} \cdot d_G(u,v)$$

$$= 16 \cdot d_G(u,v) \cdot H_n = O(\log n) \cdot d_G(u,v),$$

where H_n is the n-th harmonic number. □

2.4 Removing Steiner Nodes

Given a dominating tree T of a graph G, the question arises if we can remove the steiner nodes from T without incurring too much distortion. Gupta [17] showed how to create a tree T^* with $V(T^*) = V(G)$ with only a constant factor additional distortion. Note that T^* is not necessarily a spanning tree of G, i.e. T^* may use edges that are not in $E(G)$, but it is a dominating tree of G.

Theorem 2.4.1 (Gupta [17])**.** *Given a dominating tree $T = (V(T), E(T), w_T)$ of a weighted graph $G = (V(G), E(T), w_G)$, there exists a tree $T^* = (V(T^*) = V(G), E(T^*), w_{T^*})$ such that for any two nodes $u, v \in V(G)$, we have*

$$d_T(u,v) \leq d_{T^*}(u,v) \leq 8 d_T(u,v).$$

Moreover, $w_{T^*}(T^*) \leq 4 \cdot w_T(T)$ and T^* can be obtained from T in polynomial time.

We skip the proof for the bound on the weight of the tree T^*. We provide the algorithm and prove the essential case where the leaves of T are precisely $V(G)$. The more general case follows easily. For a tree T, let $L(T)$ denote the leaves of T.

Lemma 2.4.2. *Given a weighted tree $T = (V(T), E(T), w_T)$, there exists a tree $T^* = (V(T^*) = L(T), E(T^*), w_{T^*})$ such that for any $u, v \in L(T)$, we have*

$$d_T(u, v) \leq d_{T^*}(u, v) \leq 8d_T(u, v).$$

Proof: The following proof is a little different from the paper. Note that the set of nodes $V(T) \setminus L(T)$ are the steiner nodes in T. If T has no steiner nodes, then T is a tree on just two nodes (i.e. a single edge) and we are done. So consider the case that T has at least one steiner node. We assume that T is rooted at an arbitrary steiner node r. As we proceed, we create new trees. In these trees, there may be a steiner node which is of degree 1. We denote by $p(v)$ the parent of a node v. We call a tree *clean* if it has no steiner nodes. WLOG we assume that the distances in T are unique (ties can be broken arbitrarily).

For a node $v \in V(T)$, let $C(v)$ denote the (unique) closest leaf (distinct from v) from v in the subtree of v. Let $h(v) = d_T(v, C(v))$.

The following recursive procedure takes as input a tree T rooted at r (possibly of degree 1) and returns a clean tree T^* on the leaves of T (distinct from r). If T has only one leaf $C(r)$, then the procedure returns the tree on the single node. Otherwise, we proceed as follows. Let $T_1, \ldots, T_k$ be the maximal subtrees of T such that for any $v \in V(T_i)$, for $1 \leq i \leq k$, $d_T(r, v) \geq h(r)/2$. Let these be rooted at $r_1, \ldots, r_k$. Assume that $d_T(r, r_i) = h(r)/2$, for $1 \leq i \leq k$. If not, then we can subdivide the edge between r_i and $p(r_i)$ to add a new steiner node to T (see Figure 2–3). Recursively invoke the algorithm on $T_1, \ldots, T_k$ to obtain clean trees $T_1^*, \ldots, T_k^*$. WLOG assume that $C(r)$ is in T_1, i.e. $C(r) = C(r_1)$. Obtain the clean tree T^* by adding an edge

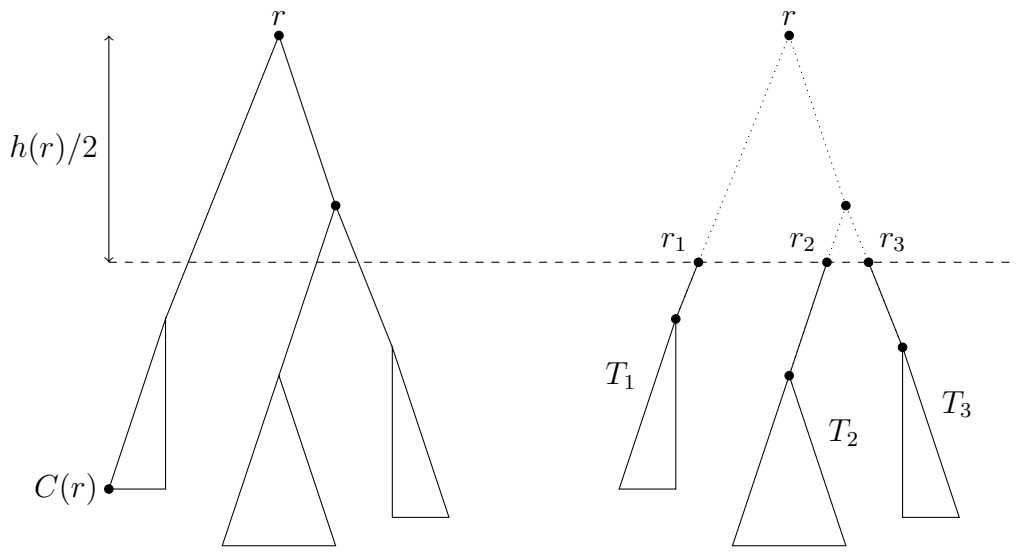


Figure 2–3: Removing Steiner Nodes: obtaining subtrees for recursion.

between $C(r) \in T_1^*$ and $C(r_i) \in T_i^*$, for each $i \in \{2, \ldots, k\}$. We assign this edge a weight of $d_T(C(r), C(r_i))$.

Note that the assumption that $d_T(r, r_i) = h(r)/2$, for $1 \leq i \leq k$, may require that nodes be added in each recursion. However, in each recursion, either the number of nodes is reduced or the distances from the root to the leaves decrease. Thus, the algorithm does terminate. Note also that when an edge is added to T^*, its weight is set to be the distance between its endpoints in T. This implies that

$$w_{T^*}(e) = d_T(u, v) \quad \forall e = uv \in E(T^*). \tag{2.1}$$

We now prove several claims about the above algorithm to bound the distances in the cleaned tree.

Claim 2.4.3. *For an input tree T rooted at r, the algorithm outputs a clean tree T^* on $L(T) \setminus \{r\}$ such that for any $u, v \in L(T) \setminus \{r\}$, we have*

$$d_{T^*}(u, v) \geq d_T(u, v).$$

Proof: For $u, v \in L(T) \setminus \{r\}$, let P_{uv} be the unique uv-path in T^*. We have

$$d_{T^*}(u, v) = \sum_{e=xy \in P_{uv}} w_{T^*}(e) = \sum_{e=xy \in P_{uv}} d_T(x, y) \geq d_T(u, v),$$

where the second equality follows from Equation (2.1). □

Claim 2.4.4. *For an input tree T rooted at r, let $T_1, \ldots, T_k$ be subtrees of T rooted at $r_1, \ldots, r_k$ as obtained during the recursive step of the algorithm. Then, for $1 \leq i \leq k$, we have*

$$d_T(C(r_1), C(r_i)) \leq 4h(r_i).$$

Proof: Note that for any $1 \leq i \leq k$, $d_T(r, r_1) = d_T(r, r_i) = h(r)/2 = h(r_1) \leq h(r_i)$. Thus, we have

$$\begin{aligned} d_T(C(r_1), C(r_i)) &\leq d_T(C(r_1), r) + d_T(r, C(r_i)) \\ &= h(r) + d_T(r, r_i) + d_T(r_i, C(r_i)) \\ &= 3h(r_1) + h(r_i) \leq 4h(r_i), \end{aligned}$$

where the first equality follows from the fact that $d_T(C(r_1), r) = h(r)$ and $d_T(r, C(r_i)) = d_T(r, r_i) + d_T(r_i, C(r_i))$. □

Claim 2.4.5. *For an input tree T rooted at r, the algorithm outputs a clean tree T^* on $L(T) \setminus \{r\}$ such that for any $u \in L(T) \setminus \{r\}$, we have*

$$d_{T^*}(u, C(r)) \leq 8d_T(u, r) - 4h(r).$$

Proof: The proof follows by induction on the size of T, i.e. $|V(T)|$. Since there must be at least one steiner node (the root node r), the base case is a tree on two nodes, r and $C(r) = u$. Thus, $h(r) = d_T(r, u)$ and so $0 = d_{T^*}(u, C(r)) \leq 8d_T(u, r) - 4h(r)$. For the inductive step, let $T_1, \ldots, T_k$ be subtrees of T rooted at $r_1, \ldots, r_k$ as obtained during the recursive step of the algorithm. Let T_i^* be the clean tree output by the algorithm with input T_i. Assume that the inequality is satisfied for T_i and T_i^*, for $1 \leq i \leq k$. Let $u \in T_i$. Note that by Equation (2.1) $d_{T^*}(C(r_i), C(r_1)) = d_T(C(r_i), C(r_1))$. We have

$$\begin{aligned} d_{T^*}(u, C(r)) &= d_{T^*}(u, C(r_i)) + d_{T^*}(C(r_i), C(r_1)) \\ &\leq d_{T^*}(u, C(r_i)) + 4h(r_i), \end{aligned}$$

where the last inequality uses Claim 2.4.4. This inequality and the inductive hypothesis imply that

$$\begin{aligned} d_{T^*}(u, C(r)) &\leq d_{T^*}(u, C(r_i)) + 4h(r_i) \\ &\leq 8d_T(u, r_i) - 4h(r_i) + 4h(r_i) \\ &= 8d_T(u, r_i) \\ &= 8(d_T(u, r) - d_T(r, r_i)) \\ &= 8d_T(u, r) - 4h(r), \end{aligned}$$

where the last equality follows from the fact that $d_T(r, r_i) = h(r)/2$. □

Claim 2.4.6. *For an input tree T rooted at r, the algorithm outputs a clean tree T^* on $L(T) \setminus \{r\}$ such that for any $u, v \in L(T) \setminus \{r\}$, we have*

$$d_{T^*}(u, v) \leq 8d_T(u, v).$$

Proof: We again proceed by induction on the size of T. The base case is a tree with a single leaf which immediately satisfies the inequality. For the inductive step, let $T_1, \ldots, T_k$ be subtrees of T rooted at $r_1, \ldots, r_k$ as obtained during the recursive step of the algorithm. Let T_i^* be the clean tree output by the algorithm with input T_i. Assume that the inequality is satisfied for T_i and T_i^*, for $1 \leq i \leq k$. If u and v are both in the same subtree T_i, then we are done by induction. So let's assume that u and v are in different subtrees T_i and T_j respectively. Note that by Equation (2.1) $d_{T^*}(C(r_i), C(r_1)) = d_T(C(r_i), C(r_1))$. We have

$$\begin{aligned} d_{T^*}(u, v) &= d_{T^*}(u, C(r_i)) + d_{T^*}(C(r_i), C(r_1)) + d_{T^*}(C(r_1), C(r_j)) + d_{T^*}(C(r_j), v) \\ &\leq d_{T^*}(u, C(r_i)) + 4h(r_i) + 4h(r_j) + d_{T^*}(C(r_j), v), \end{aligned}$$

where the inequality follows from Claim 2.4.4. Note that by Claim 2.4.5, for $1 \leq i \leq k$, $d_{T^*}(u, C(r_i)) \leq 8d_T(u, r_i) - 4h(r_i)$. With the above inequality, this implies that

$$\begin{aligned} d_{T^*}(u, v) &\leq 8d_T(u, r_i) - 4h(r_i) + 4h(r_i) + 4h(r_j) + 8d_T(v, r_j) - 4h(r_j) \\ &= 8d_T(u, r_i) + 8d_T(v, r_j) \\ &\leq 8d_T(u, v), \end{aligned}$$

where the last inequality uses the fact that $d_T(u,v) \geq d_T(u,r_i) + d_T(v,r_j)$ because u and v are in different subtrees. □

The lemma follows by using Claim 2.4.3 for the lower bound and Claim 2.4.6 for the upper bound on the distances. □

2.5 Combinatorial $\Omega(n)$ Lower Bound for a (Deterministic) Dominating Tree Embedding

We now present the combinatorial proof of the $\Omega(n)$ lower bound on the distortion of a (deterministic) dominating tree embedding of the unit-weight cycle by Gupta [17].

Theorem 2.5.1 (Gupta [17])**.** *Any dominating tree of a unit-weight cycle has distortion* $\Omega(n)$.

We consider the simpler case where the tree has no steiner nodes. The above theorem will then follow by applying Theorem 2.4.1 (note that the constant will not be the same as Theorem 2.1.6 by Rabinovich and Raz [23]).

Lemma 2.5.2. *Any dominating tree* T *of an* n*-node unit-weight cycle* C_n *with* $V(T) = V(C_n)$ *has distortion at least* $n-1$.

Proof: Let $\mathcal{T}$ be the set of all dominating trees, with no steiner nodes, that achieve the minimum distortion. Let $T \in \mathcal{T}$ be such that it has the minimum total weight, i.e. $w_T(T)$, in $\mathcal{T}$. We can assume that if $e = uv \in E(T)$, then $w_T(e) = d_{C_n}(u,v)$ (since otherwise we can reduce the edge lengths and the distortion can only go down). Let $V = \{0, 1, \ldots, n-1\}$ and assume that all additions and subtractions are modulo n. For every node i we define two *semicircles* of i as the sets $S(i) = \{i+1, i+2, \ldots, i+\lfloor n/2 \rfloor\}$ and $S'(i) = V \setminus (S(i) \cup \{i\})$.

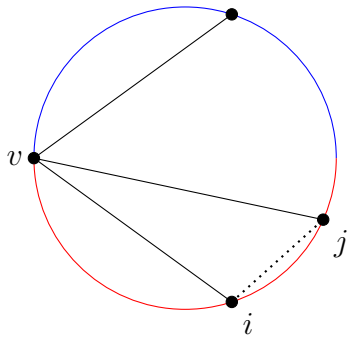


Figure 2–4: Proof of Claim 2.5.3. The two colours correspond to the two semicircles of v.

Claim 2.5.3. *For any node $v \in V(C_n)$, no semicircle of v contains more than one neighbour in T of v.*

Proof: Let us assume, for contradiction, that some node v has two neighbours in T, i and j, that lie in the same semicircle. WLOG assume that $d_{C_n}(v,i) \leq d_{C_n}(v,j)$ (see Figure 2–4). Now, $T - vj + ij$ is still a tree. By assigning ij the weight $d_{C_n}(i,j)$, we can ensure that the distortion does not increase. However,

$$d_{C_n}(v,j) = d_{C_n}(v,i) + d_{C_n}(i,j)$$

since i and j are in the same semi-circle (and thus the path in C_n from v to j passes through i). This implies that the total weight of the tree has gone down since

$$d_{C_n}(i,j) - d_{C_n}(v,j) = -d_{C_n}(v,i) < 0.$$

This contradicts the initial assumption that T has minimum total weight. □

Claim 2.5.4. *T is a path.*

Proof: This follows as a corollary of Claim 2.5.3 since every node has degree in T of at most 2. $\square$

Now, there must exist at least one edge $ij \in E(C_n)$ such that $ij \notin E(T)$ (since T can not have all the edges of the cycle). The author of [17] briefly says that $d_T(i,j)$ must be at least $n-1$. We present a more formal argument as follows.

Claim 2.5.5. *For an edge $ij \in E(C_n)$, such that $ij \notin E(T)$, we have $d_T(i,j) \geq n-1$.*

Proof: WLOG assume that $j = i+1$ and hence $j \in S(i)$. Let the unique ij-path in T be given by the sequence $i = i_0, i_1, \ldots, i_k = j$. Consider the case where $i_1 \in S'(i)$ (resp $i_1 \in S(i)$). We use P_{ij} to denote the path in C_n given by the sequence $i, i-1, \ldots, j$ (resp $i, i+1, \ldots, j$). Note that for two nodes $u, v \in V(C_n)$, P_{uv} is a shortest uv-path in C_n if $v \in S'(u)$ (resp $v \in S(u)$). Note also that, by construction, $u \in S(v)$ iff $v \in S'(u)$. We have $\bigcup_{\ell=1}^{k} P_{i_{\ell-1}i_\ell} = \{i, i-1, \ldots, i-(n-1) = j\}$ (resp $\{i, i+1, \ldots, i+(n+1) = j\}$).

We show, by induction on ℓ, that $i_\ell \in S'(i_{\ell-1})$ (resp $i_\ell \in S(i_{\ell-1})$) for $0 < \ell \leq k$. The base case for $\ell = 1$ follows from the initial assumption that $i_1 \in S'(i)$ (resp $i_1 \in S(i)$). For the inductive step, we have, by the inductive hypothesis, that $i_{\ell-1} \in S'(i_{\ell-2})$ (resp $i_{\ell-1} \in S(i_{\ell-2})$). Thus, $i_{\ell-2} \in S(i_{\ell-1})$ and so $i_\ell \in S'(i_{\ell-1})$ by Claim 2.5.3 (resp $i_{\ell-2} \in S'(i_{\ell-1})$ and $i_\ell \in S(i_{\ell-1})$).

Thus, for $0 < \ell \leq k$, $P_{i_{\ell-1}i_\ell}$ is a shortest $i_{\ell-1}i_\ell$-path in C_n. Therefore,

$$d_T(i, i+1) = \sum_{\ell=1}^{k} d_T(i_{\ell-1}, i_\ell) \geq \sum_{\ell=1}^{k} d_{C_n}(i_{\ell-1}, i_\ell) = |\bigcup_{\ell=1}^{k} P_{i_{\ell-1}i_\ell}| - 1,$$

where the inequality is because T is a dominating tree of C_n. The claim follows since $|\bigcup_{\ell=1}^{k} P_{i_{\ell-1}i_\ell}| = n$ (resp $n+2$). $\square$

The lemma follows by Claim 2.5.5 since there is at least one edge $ij \in E(C_n)$ such that $ij \notin E(T)$. □

2.6 Derandomizing Probabilistic Embeddings

We have so far considered probabilistic embeddings where the expected stretch of any edge is small. We can similarly consider deterministic trees such that the *average stretch* is small. As with probabilistic embeddings, we may consider either subtrees or dominating trees. Formally, we define average stretch of a (spanning or dominating) tree T of a graph G as

$$\text{av\_str}_T(G) = \frac{1}{|E(G)|} \sum_{uv \in E(G)} \frac{d_T(u,v)}{d_G(u,v)}.$$

For a multigraph, the average stretch is weighted according to the edge multiplicies. Recall from Section 1.2 that for a multigraph G each edge $uv \in E(G)$ has edge multiplicity $m_{uv} > 0$ and $M = \sum_{uv \in E(G)} m_{uv}$. The average stretch of a (spanning or dominating) tree T of a multigraph G is defined as

$$\text{av\_str}_T(G) = \frac{1}{M} \sum_{uv \in E(G)} m_{uv} \frac{d_T(u,v)}{d_G(u,v)}.$$

By employing standard derandomization techniques ([4], [25]), algorithms for probabilistic embeddings can be derandomized, using conditional probabilities, to get low average stretch trees for multigraphs. We present the derandomized procedure of

Fakcharoenphol et al. [14, 27] to get a dominating tree with $O(\log n)$ average stretch[2] for a general multigraph. Fakcharoenphol et al.'s derandomized procedure follows a region growing method instead of the standard techniques referred to above.

Theorem 2.6.1. *For any multigraph G, there exists a single dominating tree T such that $av\_str_T(G) = O(\log n)$.*

Proof: WLOG we assume that the graph G is connected with $w(e) > 1$, for every edge $e \in E(G)$. Let $B_G(z, \rho)$ denote a *ball* of radius ρ centered at node z in G, i.e. the set of nodes in G at distance at most ρ from z. The *volume* of this ball, denoted by $\text{vol}_G(z, \rho)$, is defined as follows. Each edge $e = uv \in E(G)$ with both endpoints in $B_G(z, \rho)$ contributes m_{uv} to $\text{vol}_G(z, \rho)$. An edge $e = uv \in E(G)$ with only one endpoint, say u, in the ball contributes $\frac{m_{uv}}{w(e)} \cdot \big(\rho - d_G(z, u)\big)$ to $\text{vol}_G(z, \rho)$. Intuitively, one can view each edge $e = uv \in E(G)$ as a pipe with length $w(e)$ and cross sectional area $\frac{m_{uv}}{w(e)}$ ([27]). We add an additional volume of $\frac{M}{n}$ on each node. This will help us bound the total stretch later. Let $\text{vol}(G)$ denote the volume of the entire graph G. We have

$$\text{vol}(G) = \sum_{uv \in E(G)} m_{uv} + \sum_{u \in V(G)} \frac{M}{n} = 2M. \tag{2.2}$$

[2]Fakcharoenphol et al. define the average stretch to be $\text{av\_str}_T(G) = \frac{\sum_{uv \in E(G)} m_{uv} d_T(u,v)}{\sum_{uv \in E(G)} m_{uv} d_G(u,v)}$ which is different than our definition. In fact, this is similar to our definition of communication cost later in Chapter 4. As such their procedure is modified slightly to get the same $O(\log n)$ upper bound.

Let $A_G(z,\rho) = \sum_{u \in B_G(z,\rho),\ v \notin B_G(z,\rho)} \frac{m_{uv}}{w(e)}$. Note that

$$\frac{d\mathrm{vol}_G(z,\rho)}{d\rho} \geq A_G(z,\rho),$$

for any z and ρ. For a set $S \subseteq V$, we may write $B_S(\cdot,\cdot)$ instead of $B_{G[S]}(\cdot,\cdot)$ when G is understood. Similarly, we may write $A_S(\cdot,\cdot)$ instead of $A_{G[S]}(\cdot,\cdot)$ and $\mathrm{vol}_S(\cdot,\cdot)$ instead of $\mathrm{vol}_{G[S]}(\cdot,\cdot)$. We drop the subscript when it is clear from the context.

We first show that for any $z \in V(G)$ there exists a $\rho_i : 2^{i-1} \leq \rho_i < 2^i$ such that

$$\frac{A(z,\rho_i)}{\mathrm{vol}(z,\rho_i)} \leq 2^{-(i-1)} \cdot \ln \frac{\mathrm{vol}(z,2^i)}{\mathrm{vol}(z,2^{i-1})}. \tag{2.3}$$

Suppose, for contradiction, that $\frac{A(z,\rho_i)}{\mathrm{vol}(z,\rho_i)} > 2^{-(i-1)} \ln \frac{\mathrm{vol}(z,2^i)}{\mathrm{vol}(z,2^{i-1})}$ for all $2^{i-1} \leq \rho_i < 2^i$. Since $\int \frac{1}{\mathrm{vol}(z,\rho_i)} d\mathrm{vol}(z,\rho_i) = \ln \mathrm{vol}(z,\rho_i)$, we have that

$$\begin{aligned}
\ln \frac{\mathrm{vol}(z,2^i)}{\mathrm{vol}(z,2^{i-1})} &= \int_{2^{i-1}}^{2^i} \frac{1}{\mathrm{vol}(z,\rho_i)} \frac{d\mathrm{vol}(z,\rho_i)}{d\rho_i} d\rho_i \\
&\geq \int_{2^{i-1}}^{2^i} \frac{1}{\mathrm{vol}(z,\rho_i)} A(z,\rho_i) d\rho_i \\
&> \int_{2^{i-1}}^{2^i} 2^{-(i-1)} \ln \frac{\mathrm{vol}(z,2^i)}{\mathrm{vol}(z,2^{i-1})} d\rho_i \\
&= \ln \frac{\mathrm{vol}(z,2^i)}{\mathrm{vol}(z,2^{i-1})},
\end{aligned}$$

which is a contradiction.

Let Δ be the diameter of G and let $\gamma = \lceil \log_2 \Delta \rceil$. Recall (from Section 2.3) that, for a graph G, a laminar family $\mathcal{F} \subseteq 2^{V(G)}$ is associated with a rooted tree T. Each node in T is at some *level* in the tree. The root is at level γ. The level decreases with increasing distance from the root. We say that a cluster $S \in \mathcal{F}$ is at level $i+1$ if its associated node v_S in T is at level $i+1$. Fakcharoenphol et al. ensure that a

cluster at level $i+1$ has radius at most 2^{i+1}. Note that, since the edge weights are strictly greater than 1, the lowest level in T is level 0. As in Section 2.3, an edge from a level $i+1$ node in T to a level i node in T is assigned a weight of 2^{i+1}. This ensures that T is a dominating tree (Section 2.3).

Similar to Section 2.3, we recursively grow regions that define a laminar family as follows. Initially, $V(G)$ is set to be at level γ. For each level $i+1$, we create child clusters at level i as follows. Consider a cluster S at level $i+1$ and the subgraph $G[S]$. Select a node $z \in S$ such that $\text{vol}_S(z, 2^{i-1})$ is maximized and grow a ball, in $G[S]$, of radius ρ_i (selected as above: $2^{i-1} \leq \rho_i < 2^i$ such that Equation (2.3) holds). Set $S_1 = B_S(z, \rho_i)$ and repeat on $S \setminus S_1$ to partition S into child clusters $S_1, \ldots, S_k$. The radius of each child cluster is at most 2^i and hence its diameter is at most 2^{i+1}, as desired for a cluster at level i.

For analysis, we place a *credit* of $\sum_{e \in \delta(u)} m_e + \frac{M}{n}$ on each node u. Let credit(S) be the sum of credits in a cluster S. The total credits in the graph are $\sum_{u \in V} \left(\sum_{e \in \delta(u)} m_e + \frac{M}{n} \right) = 3M$. Similarly, a cluster can not have less credits than its volume. Thus, for any cluster S, using $\text{vol}_G(S)$ to denote the volume of S in graph G, we have

$$3M \geq \text{credit}(S) \geq \text{vol}_G(S) \geq \text{vol}_H(S), \tag{2.4}$$

for any subgraph H of G.

We now bound the average stretch of the resulting tree. Consider one step of the algorithm, where we have clusters at level $i+1$ and new ones are being created at level i. Recall that an edge from a level $i+1$ node in T to a level i node in T is assigned a weight of 2^{i+1}. Consider an edge $uv \in E(G)$ such that both u and v are

in the same cluster S_{i+1} at level $i+1$ but are separated into two different clusters, S_i and S_i', at level i. Then $d_T(u,v) \leq \sum_{j=0}^{i} 2 \times 2^{j+1} \leq 2^{i+3}$. So, the edge uv contributes at most $\frac{m_{uv}}{w(e)} \cdot 2^{i+3}$ to the total stretch. We charge this to clusters S_i and S_i' equally so that each of them gets charged $\frac{m_{uv}}{w(e)} \cdot 2^{i+2}$.

Consider a node $u \in V(G)$. For each level $i \in \{0, \ldots, \gamma\}$, let u be in cluster S_i with center z_i and radius ρ_i. The total stretch charged to each S_i is given by

$$\sum_{u \in S_i,\ v \in S_{i+1} \setminus S_i} \frac{m_{uv}}{w(e)} \cdot 2^{i+2} = A_{S_{i+1}}(z_i, \rho_i) \cdot 2^{i+2}.$$

This is shared equally by the credits in S_i. Note that, by choice of z_i, we have

$$\mathrm{vol}_{S_{i+1}}(z_i, 2^{i-1}) \geq \mathrm{vol}_{S_{i+1}}(v, 2^{i-1}) \quad \forall v \in S_i \tag{2.5}$$

Now, at level i, each unit of credit at u gets charged

$$\frac{A_{S_{i+1}}(z_i, \rho_i) \cdot 2^{i+2}}{\mathrm{credit}(S_i)} \leq \frac{A_{S_{i+1}}(z_i, \rho_i)}{\mathrm{vol}_{S_{i+1}}(z_i, \rho_i)} \cdot 2^{i+2} \leq 8 \cdot \ln \frac{\mathrm{vol}_{S_{i+1}}(z_i, 2^i)}{\mathrm{vol}_{S_{i+1}}(z_i, 2^{i-1})},$$

where we have used Equation (2.4) for the first inequality and the property of ρ_i given by Equation (2.3) for the second inequality. By summing over i, the upper bound on the total stretch charged to one unit of credit at u is

$$\begin{aligned}
\sum_{i=0}^{\gamma-1} 8 \cdot \ln \frac{\mathrm{vol}_{S_{i+1}}(z_i, 2^i)}{\mathrm{vol}_{S_{i+1}}(z_i, 2^{i-1})} &= 8 \cdot \ln \prod_{i=0}^{\gamma-1} \frac{\mathrm{vol}_{S_{i+1}}(z_i, 2^i)}{\mathrm{vol}_{S_{i+1}}(z_i, 2^{i-1})} \\
&= 8 \cdot \ln \Big(\frac{\mathrm{vol}_{S_\gamma}(z_{\gamma-1}, 2^{\gamma-1})}{\mathrm{vol}_{S_1}(z_0, 2^0)} \cdot \prod_{i=1}^{\gamma-1} \frac{\mathrm{vol}_{S_i}(z_{i-1}, 2^{i-1})}{\mathrm{vol}_{S_{i+1}}(z_i, 2^{i-1})} \Big) \\
&\leq 8 \cdot \ln \Big(\frac{\mathrm{vol}_{S_\gamma}(z_{\gamma-1}, 2^{\gamma-1})}{\mathrm{vol}_{S_1}(z_0, 2^0)} \cdot \prod_{i=1}^{\gamma-1} \frac{\mathrm{vol}_{S_{i+1}}(z_{i-1}, 2^{i-1})}{\mathrm{vol}_{S_{i+1}}(z_i, 2^{i-1})} \Big)
\end{aligned}$$

$$\leq 8 \cdot \ln \frac{\mathrm{vol}(G)}{\mathrm{vol}_{S_1}(z_0, 1)},$$

where the first inequality follows by the fact that $\mathrm{vol}_{S_i}(z_{i-1}, 2^{i-1}) \leq \mathrm{vol}_{S_{i+1}}(z_{i-1}, 2^{i-1})$ and the second inequality follows by telescoping due to Equation (2.5) (since $z_{i-1} \in S_i$) and upper bounding the remaining term. Note that $\mathrm{vol}_{S_1}(z_0, 1) = \frac{M}{n}$ (the volume placed at the node z_0). By using Equation (2.2) we get an upper bound of

$$8 \cdot \ln \frac{2M}{M/n} = 8 \cdot \ln(2n)$$

for total stretch charged to one unit of credit at any node. The total stretch is then at most

$$\mathrm{credit}(V(G)) \cdot 4 \cdot \ln(2n) \leq 12 \cdot M \cdot \ln(2n) = O(M \log n),$$

where we have used Equation (2.4) for the inequality. Thus, the average stretch of the tree is $O(\log n)$. □

CHAPTER 3
Embeddings Using Trees in the Graph

3.1 Spanning Tree Embeddings for General Graphs

So far we have discussed embeddings that allow steiner nodes (and edges). This leads to a tight $\Theta(\log n)$-probabilistic embedding [14]. However, if the trees are restricted to be subgraphs of the original graph G, then there is a $O(\log \log n)$ gap between the best upper and lower bounds.

We now give a brief historical excursion of results in this area. Alon et al. [3] were the first to consider probabilistic embeddings into spanning trees (although it was only implicit in their construction). They gave a $\Omega(\log n)$ lower bound and conjectured that this lower bound is tight. They also showed, using the von Neumann minimax principle of game theory, that a weighted graph G allows an α-probabilistic embedding into a distribution over its spanning trees if and only if every multigraph obtained from G via edge-replication[1] admits a spanning tree with average stretch α.

The upper bound results for general graphs have generally been established using region growing techniques. By modifying Awerbuch's clustering algorithm ([5]), Alon et al. [3] obtained probabilistic embeddings into spanning trees with distortion

[1]Edge-replication of a graph G is an assignment of edge multiplicities $m_{uv} > 0$ for every edge $uv \in E(G)$.

$e^{O(\sqrt{\log n \log\log n})}$. Elkin et al. [11] significantly improved the upper bound to $O(\log^2 n \cdot \log\log n)$ using low cost *star decompositions*. A star decomposition partitions the node set of a graph G into $\{V_0, V_1, \ldots, V_k\}$ which form a star where V_0 is the central cluster and each V_i, for $1 \leq i \leq k$, is connected to the central cluster by a single edge. By recursively invoking their algorithm for low cost star decompositions on each cluster, Elken et al. grow a tree with the required average stretch. Abraham et al. [1] improved the star decomposition framework to obtain probabilistic embeddings with distortion $O(\log n \cdot \log\log n \cdot (\log\log\log n)^3)$. By using a different decomposition, called the *petal decomposition*, Abraham et al. [2] construct a spanning tree with average distortion $O(\log n \cdot \log\log n)$. In comparison to the star decomposition, in a petal decomposition the clusters are grown differently and are connected in a tree like manner rooted at V_0. As far as we know, this is the best known bound for spanning tree embeddings of general graphs. This leaves a gap of $O(\log\log n)$ between the best upper and lower bounds.

Theorem 3.1.1 (Abraham et al. [2])**.** *For any weighted graph $G = (V, E, w)$ there exists a spanning tree T such that $av\_str_T(G) = O(\log n \cdot \log\log n)$. Moreover, T can be found in polynomial time.*

We now present results on some special classes of graphs. In some cases, not only can we obtain distributions over subtrees, but the logarithmic distortion bound can be improved as well. Recall that the probabilistic embeddings can be derandomized to obtain low average stretch trees using standard techniques (Section 2.6) even when subtrees are required.

3.2 Embedding Outerplanar Graphs into Spanning Trees

In this section, we reproduce a proof of Gupta et al. [18] that shows that any outerplanar graph has a constant distortion probabilistic embedding into its spanning trees.

Theorem 3.2.1 (Gupta et al. [18])**.** *Any outerplanar graph $G = (V, E, w)$ can be probabilistically embedded into a distribution over its spanning trees with distortion at most 8.*

It is clearly sufficient to find spanning trees on the biconnected components of the graph. So, WLOG we assume that the outerplanar graph is biconnected. We use the ear decomposition, $\langle P_0, P_1, \ldots, P_\ell \rangle$, of an outerplanar graph G (see Section 1.4.1). A path P_i in this decomposition, connected to an edge $e_i \in E(G_{i-1})$, is called *slack* if $w(P_i) \geq 2 \cdot w(e_i)$ (that is the weight of P_i is at least twice the weight of e_i). Further, a decomposition is called *slack* if all the paths P_i in it are slack.

Gupta et al. showed that every outerplanar graph has a subgraph with distortion 2 that has a slack decomposition. We show the following lemma using a slighty different proof, but the same techniques. Note that their subgraphs are allowed to have different edge weights than the original graph.

Lemma 3.2.2. *Given an outerplanar graph $G = (V, E, w)$, there is a subgraph $H = (V, E', w)$ of G with a slack decomposition such that for any $u, v \in V$,*

$$d_G(u, v) \leq d_H(u, v) \leq 2d_G(u, v).$$

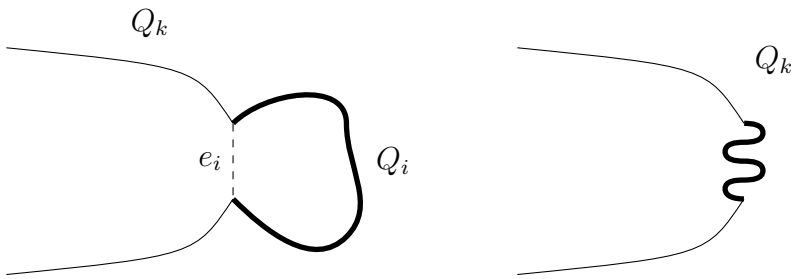


Figure 3–1: Finding subgraphs with slack decompositions. In this example, Q_i is not slack.

Proof: Let $\mathcal{P} = \langle P_0, P_1, \ldots, P_\ell \rangle$ be the decomposition for G. We find a slack composition $\mathcal{Q} = \langle Q_0, Q_1, \ldots, Q_{\ell'} \rangle$ for H. We work in reverse to modify G's decomposition $\mathcal{P}$ to build $\mathcal{Q}$. We start with $i = \ell$. At each iteration we do the following. If Q_i is slack, we do nothing and continue to the next iteration ($i = i - 1$). If Q_i is not slack, then it is connected to the endpoints of an edge, say e_i, and the weight of Q_i is less than twice the weight of e_i. The edge e_i lies on some earlier path Q_k with $k < i$. Modify Q_k by replacing e_i with the entire path Q_i (see Figure 3–1). Remove Q_i from the composition and continue to the next iteration ($i = i - 1$). The effect is to merge the path Q_i with the earlier path Q_k. At the end of the procedure, reindex Q_i's still in the composition from 1 to ℓ'. Note that we can argue inductively that for any k we have $Q_k \subseteq \cup_{i \geq k} P_i$.

At the end of the above procedure, the graph H, defined by the composition $\langle Q_0, Q_1, \ldots, Q_{\ell'} \rangle$, is a subgraph of G. Moreover, the composition is slack by construction. So, it remains to show that $d_H(u, v) \leq 2 \cdot d_G(u, v)$, for any $u, v \in V$. Recall that each P_i is connected to the endpoints of some edge $e_i = u_i v_i \in E(G_{i-1})$. It is sufficient to show that for any such edge $d_H(u_i, v_i) \leq 2 \cdot d_G(u_i, v_i)$ since all other edges

are included in $E(H)$. Note that, before the reindex, endpoints of Q_i correspond to the endpoints of P_i for each i. For some i, if e_i was never replaced by another path then $d_H(u_i, v_i) = d_G(u_i, v_i)$ and we are done. So assume that e_i was replaced by a path Q_i. So far $d_H(u_i, v_i) \leq 2 \cdot d_G(u_i, v_i)$ since Q_i was not slack. For $d_H(u_i, v_i)$ to be greater than $2 \cdot d_G(u_i, v_i)$, it must be the case that some edge $e_j \in Q_i$ is replaced by some path Q_j such that $j < i$. Since $Q_i \subseteq \bigcup_{i' \geq i} P_{i'}$, we have that $e_j \in P_{i'}$ for some $i' \geq i > j$. Moreover, since the endpoints of Q_j correspond to the endpoints of P_j, we have that P_j is attached to the endpoints of $e_j \in P_{i'}$ such that $j < i'$. This violates the property of the ear decomposition that P_j is connected to an edge $e_j \in E(G_{j-1})$ (note that $E(G_{j-1}) = \bigcup_{j' < j} P_{j'}$). Thus, $d_H(u_i, v_i) \leq 2 \cdot d_G(u_i, v_i)$. □

We now have the following lemma by Gupta et al. [18].

Lemma 3.2.3 (Gupta et al. [18])**.** *Given an outerplanar graph G with a slack decomposition, G can be probabilistically embedded into a distribution over its spanning trees with distortion at most* 4*.*

Proof: The proof is constructive following the decomposition. We construct a random spanning tree T_i of G_i from a random spanning tree T_{i-1} of G_{i-1} by setting $T_i = T_{i-1} \cup \{P_i \backslash e\}$, where e is an edge of P_i picked with probability $p(e) = w(e)/w(P_i)$ (see Figure 3–2). We want to show that at each step $d_{T_i}(x, y) \leq 4 d_{G_i}(x, y)$, for each edge $xy \in G_i$. This can be shown by induction on i.

For the base case, note that G_0 is a cycle and so we can use Lemma 2.1.4. For the inductive step, assume that $d_{T_{i-1}}(x, y) \leq 4 d_{G_{i-1}}(x, y)$, for all $xy \in E(G_{i-1})$. Now, at step i we have not changed any of the paths in T_{i-1}. Thus, for any edge

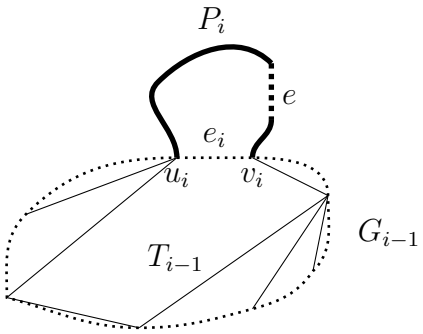


Figure 3–2: Constructing a random spanning tree T_i of G_i from a random spanning tree T_{i-1} of G_{i-1}.

$xy \in E(G_{i-1})$, we have

$$\mathbb{E}[d_{T_i}(x,y)] = \mathbb{E}[d_{T_{i-1}}(x,y)] \leq 4d_{G_{i-1}}(x,y) = 4d_{G_i}(x,y),$$

where the last equality follows from the assumption that the edge xy is a shortest xy-path.

Let P_i be connected to the endpoints of an edge $e_i = u_iv_i \in E(G_{i-1})$. For any edge $e = xy \in P_i$, if it is removed, then $d_{T_i}(x,y) = w(P_i) - w(e) + d_{T_i}(u_i, v_i)$. Thus, we have

$$\begin{aligned}\mathbb{E}[d_{T_i}(x,y)] &= p(e) \cdot (w(P_i) - w(e) + \mathbb{E}[d_{T_i}(u_i,v_i)]) + (1-p(e)) \cdot w(e) \\ &\leq \frac{w(e)}{w(P_i)} \cdot \big(w(P_i) - w(e) + 4w(e_i)\big) + \frac{w(P_i) - w(e)}{w(P_i)} \cdot w(e),\end{aligned}$$

where the inequality follows from the induction hypothesis. Rearranging terms we get

$$\mathbb{E}[d_{T_i}(x,y)] \leq \Big(2\frac{w(P_i) - w(e)}{w(P_i)} + 4\frac{w(e_i)}{w(P_i)}\Big)w(e)$$

$$\leq \Big(2\frac{w(P_i) - w(e)}{w(P_i)} + 2\Big)w(e)$$
$$\leq (2+2)w(e) = 4d_{G_i}(x,y),$$

where the second inequality follows from the slack decomposition, i.e. $w(P_i) \geq 2w(e_i)$. □

Theorem 3.2.1 is now immediate.

3.3 k-Outerplanar Graphs

Chekuri et al. [9] extended the above procedure on outerplanar graphs to show that every k-outerplanar graph can be probabilistically embedded into a distribution over dominating trees with distortion exponential in k (but independent of n). Yuval Emek [12] enhanced the result by restricting the trees to be the subtrees of the original graph. Both results are based on standard "onion peeling" techniques where the nodes on the unbounded face are peeled off to recursively obtain a tree for the $(k-1)$-outerplanar graph. Reinserting the missing nodes (and edges) essentially creates a *Halin* graph. The main task is then to construct low stretch trees for this graph.

Theorem 3.3.1 (Emek [12])**.** *Any k-outerplanar graph G can be probabilistically embedded into a distribution over its spanning trees with distortion c^k where c is an absolute constant.*

3.4 Embedding Series-Parallel Graphs into their Spanning Trees

Emek and Peleg [13] gave a constructive argument for an upper bound for probabilistically embedding any **unweighted** series-parallel graph into its spanning trees with distortion $O(\log n)$. This is tight due to the lower bound of Gupta et al. [18]

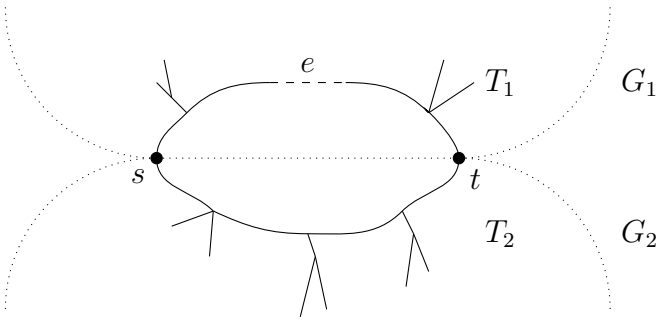


Figure 3–3: Constructing a random spanning tree T of G from a random spanning tree T_1 of G_1 and a random spanning tree T_2 of G_2. G is a parallel composition of G_1 and G_2.

presented in Section 2.2. For the weighted version the same procedure can be adapted to yield a distortion of $O(\log(n + \Delta))$ where Δ is the largest edge weight.

Emek and Peleg's [13] procedure relies on the composition of a series-parallel graph using the series and parallel operations (see Section 1.4.2). Their randomized algorithm for series-parallel graphs is essentially as follows. A random spanning tree T of G is constructed recursively. At any stage, we have that G is either a series or parallel composition of two graphs G_1 and G_2. Recursively run the algorithm on G_1 and G_2 to obtain random spanning trees T_1 and T_2 respectively. If G is a series composition of G_1 and G_2, then T is simply the series composition of T_1 and T_2. If G is a parallel composition of G_1 and G_2 with terminals s and t, then we obtain T as follows. WLOG assume that T_2 has the shorter path between s and t. Let e be an edge on the unique path between s and t in T_1, picked at random with probability proportional to $w(e)$. Obtain T from the parallel composition of T_1 and T_2 by removing the edge e (see Figure 3–3).

To upper bound the expected distortion, consider an edge $uv \in E(G)$. We say that a series/parallel operation $\mathfrak{A}$, with input graphs G_1 and G_2, *involves* an edge e if either $e \in E(G_1)$ or $e \in E(G_2)$. Let $\mathfrak{A}^1, \mathfrak{A}^2, \ldots \mathfrak{A}^\ell$ be the sequence of series/parallel operations which involve the edge uv. Let the input to the operation $\mathfrak{A}^i$ be the subgraphs G_1^i and G_2^i with the random spanning trees T_1^i and T_2^i respectively. As before, assume WLOG that T_2^i has the shorter path between the terminals. This can be defined deterministically to precisely identify T_2's before the start of the algorithm. We also assume WLOG that each operation $\mathfrak{A}^i$ is a parallel operation (since a series operation does not change the uv-path) and that the edge uv is in G_1^i (otherwise the uv-path remains the same since no edge is deleted from T_2^i). Let P_1^i and P_2^i be the paths between the terminals in T_1^i and T_2^i respectively. Let ψ^i be the (unique) uv-path in T_1^i and let χ^i be the part of that path in P_1^i (i.e $\chi^i = \psi^i \cap P_1^i$), as shown in Figure 3–4. Now, we define the variables a_i, b_i, c_i, and d_i as follows:

- $a_i = w(P_1^i \setminus \chi^i) = w(P_1^i) - w(\chi^i)$.
- $b_i = w(\chi^i)$.
- $c_i = w(\psi^i \setminus \chi^i) = w(\psi^i) - w(\chi^i)$.
- $d_i = w(P_2^i)$.

Recall that we are in the unweighted setting with $w(e) = 1$ for all edges e and thus, e.g. $w(\psi^i)$ is only counting the number of edges in the path ψ^i.

Call a path ψ^i *settled* if it does not overlap with P_1^i, i.e. $\chi^i = \emptyset$. Observe that if ψ^i is settled, then ψ^{i+1} is settled as well and if ψ^i is not settled, then ψ^{i+1} is settled only if χ^i remains connected after the random edge deletion. If ψ^{i+1} is not settled, then $\chi^{i+1} = P_2^i$. Based on these observations, we define A_i as the event that ψ^{i+1}

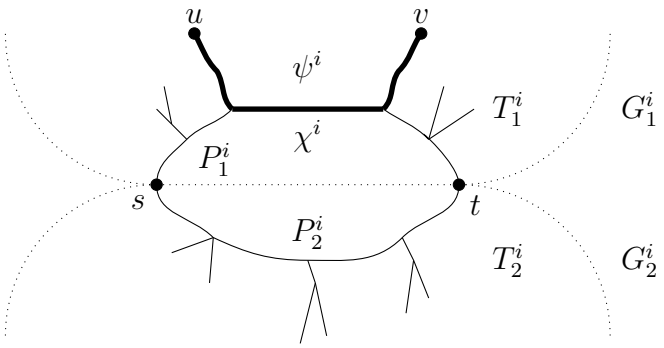


Figure 3–4: Analyzing the algorithm to construct a random spanning tree of a series-parallel graph.

is settled at operation $\mathfrak{A}^i$ and B_i as the event that ψ^{i+1} is not settled. We have $B_i = \neg A_1 \wedge \cdots \wedge \neg A_i$ and hence

$$\mathbb{P}[B_i] = \mathbb{P}[B_i \wedge B_{i-1}] = \mathbb{P}[B_i | B_{i-1}] \cdot \mathbb{P}[B_{i-1}]$$

and

$$\mathbb{P}[A_i] = \mathbb{P}[A_i \wedge B_{i-1}] = \mathbb{P}[A_i | B_{i-1}] \cdot \mathbb{P}[B_{i-1}].$$

Expanding this, one obtains

$$\mathbb{P}[A_i] = \mathbb{P}[A_i | B_{i-1}] \Big(\prod_{j=2}^{i-1} \mathbb{P}[B_j | B_{j-1}] \Big) \cdot \mathbb{P}[B_1].$$

Note that $\mathbb{P}[A_i | B_{i-1}] = \frac{a_i}{a_i + b_i}$, $\mathbb{P}[B_i | B_{i-1}] = \frac{b_i}{a_i + b_i}$, and $\mathbb{P}[B_1] = \frac{b_1}{a_1 + b_1}$. So, one has

$$\mathbb{P}[A_i] = \frac{a_i}{a_i + b_i} \Big(\prod_{j=1}^{i-1} \frac{b_j}{a_j + b_j} \Big).$$

We can use this to get the expression for the expected distance between u and v as follows.

$$\mathbb{E}[d_T(u,v)] = \Big(\prod_{i=1}^{\ell} \frac{b_i}{a_i+b_i}\Big)(a_\ell + c_\ell + d_\ell) + \sum_{i=1}^{\ell} \frac{a_i}{a_i+b_i}\Big(\prod_{j=1}^{i-1} \frac{b_j}{a_j+b_j}\Big)(b_i+c_i),$$

where the left most term corresponds to the event B_ℓ. By bounding the value of the right hand side in the above expression, Emek and Peleg obtain the following.

Theorem 3.4.1 (Emek and Peleg [13])**.** *Any unweighted series-parallel graph G can be α-probabilistically embedded into a distribution over its spanning trees with $\alpha = O(\log n)$.*

CHAPTER 4
MCST and LP Based Flow Formulations for MCST

We now turn our attention towards the *communication cost* of a graph. We are given a (symmetric) matrix D of non-negative *demands* (or *communication requirements*) for a graph G. The demand between $u, v \in V(G)$ is given by D_{uv}. For a graph H with $V(H) \supseteq V(G)$, we define a communication cost (for G)

$$\text{com-cost}_H(G) = \sum_{u,v \in V(G)} D_{uv} \cdot d_H(u, v).$$

If $H = G$, then we write com-cost(G).

We wish to find trees with low communication cost. When restricted to subtrees of the original graph G, this is the problem MCST[1]. It is clear that if a tree T has distortion at most α, then $\text{com-cost}_T(G) \leq \alpha \cdot \text{com-cost}(G)$. In fact, if there is an α-probabilistic embedding into a distribution over trees, then there is a single tree T whose $\text{com-cost}_T(G) \leq \alpha \cdot \text{com-cost}(G)$ (we discuss this in Section 4.1). If the trees are subtrees of G, then this gives an α-approximation to MCST (since com-cost(G) is a lower bound on the communication cost of the best possible tree). It is possible, however, that the best tree has distortion $O(\log n)$ yet one may still be

[1]Some people consider these problems without the restriction that the trees should be subtrees of the original graph.

able to $O(1)$-approximate (in polytime) the communication cost of a best tree. Such an algorithm would obviously need a better lower bound than the optimal distortion of tree embeddings. For example, Emek and Peleg's procedure (Section 3.4) gives an $O(\log n)$-probabilistic embedding for series-parallel graphs. However, for MCST instances on the diamond graphs with unit demands on the edges, it gives an $O(1)$-approximation due to the lower bound by Gupta et al. (Section 2.2). In this chapter, we seek such stronger approximations to MCST.

In Section 4.1, we first discuss approximating MCST by finding trees which perform well against com-cost(G). In Sections 4.2 and 4.3, we present linear programming based formulations for MCST. We also investigate their integrality gaps based on the family of diamond graphs with unit demands on edges.

4.1 An Equivalence between Communication Cost and Average Stretch when bounding relative to com-cost(G)

Given a graph G with a (symmetric) demand matrix D, consider the problem of finding a (dominating or spanning) tree T such that com-cost$_T(G) \leq \alpha \cdot$com-cost(G) for some α. It is sufficient to consider the demands only on the edges of G ([13]). This can be shown as follows. We create a new demand matrix D' such that $D'_{uv} = 0$ if $uv \notin E(G)$. Let P_{uv} be a shortest uv-path (breaking ties arbitrarily) in G. Then, we set D'_{uv} as follows for every $u, v \in V(G)$.

$$D'_{uv} = \sum_{x,y|uv \in P_{xy}} D_{xy}. \tag{4.1}$$

By definition $D'_{uv} = 0$ if $uv \notin E(G)$.

Proposition 4.1.1 ([13]). *For any subgraph H of G with demand matrix D, D' satisfies*

$$\sum_{u,v \in V(G)} D_{uv} \cdot d_H(u,v) \leq \sum_{u,v \in V(G)} D'_{uv} \cdot d_H(u,v),$$

where the inequality is tight when $H = G$. I.e., the communication cost of D in H is at most the communication cost of D' in H.

Proof: For any subgraph H of G, we have

$$\begin{aligned}
\sum_{u,v \in V(G)} D_{uv} \cdot d_H(u,v) &\leq \sum_{u,v \in V(G)} D_{uv} \sum_{xy \in P_{uv}} d_H(x,y) \\
&= \sum_{xy \in E(G)} d_H(x,y) \sum_{u,v | xy \in P_{uv}} D_{uv} \\
&= \sum_{x,y \in V(G)} d_H(x,y) \cdot D'_{xy},
\end{aligned}$$

where the only inequality is tight when $H = G$ and the last equality follows from (4.1). □

Consider a graph G with demand matrix D. Let com-cost$'_T(G)$ denote the communication cost of the tree T with the demand matrix D'. Using D', we find a low stretch tree T by some method, i.e. such that com-cost$'_T(G) \leq \alpha \cdot$ com-cost$'(G)$. Then, T will have com-cost$_T(G) \leq \alpha \cdot$ com-cost(G) since

$$\text{com-cost}_T(G) \leq \text{com-cost}'_T(G) \leq \alpha \cdot \text{com-cost}'(G) = \alpha \cdot \text{com-cost}(G),$$

where the first inequality follows from Proposition 4.1.1 and the equality follows from the second part of Proposition 4.1.1.

We now deduce an equivalence between average stretch (see Section 2.6) and communication cost as specified in the following proposition.

Proposition 4.1.2 ([22, 26])**.** *Let T be any (dominating or spanning) tree of a simple graph $G = (V, E, w)$. Then, for every demand matrix D, whose support is in $E(G)$, there exist multiplicities m_{uv}, for every $uv \in E(G)$, such that*

$$\textit{com-cost}_T(G) = \textit{av\_str}_T(G) \cdot \textit{com-cost}(G). \tag{4.2}$$

Conversely, given multiplicities defining an averge stretch problem on G, one may define a demand matrix whose support is in $E(G)$ such that Equation (4.2) holds.

Proof: Let $m_{uv} = D_{uv} \cdot d_G(u, v)$. Since $D_{uv} = 0$ if $uv \notin E(G)$, we have

$$\begin{aligned}
\text{com-cost}_T(G) &= \sum_{uv \in E(G)} D_{uv} \cdot d_T(u, v) \\
&= \sum_{uv \in E(G)} m_{uv} \cdot \frac{d_T(u, v)}{d_G(u, v)} \\
&= M \cdot \text{av_str}_T(G) \\
&= \text{av_str}_T(G) \cdot \sum_{uv \in E(G)} m_{uv} \\
&= \text{av_str}_T(G) \cdot \sum_{uv \in E(G)} D_{uv} \cdot d_G(u, v) \\
&= \text{av_str}_T(G) \cdot \text{com-cost}(G).
\end{aligned}$$

By letting $D_{uv} = \frac{m_{uv}}{d_G(u,v)}$, the other direction follows similarly. □

Corollary 4.1.3. *An algorithm to α-probabilistically embed a multigraph G into a distribution over spanning trees of G implies an α-approximation algorithm to MCST.*

Proof: Consider an MCST instance on a graph G with a demand matrix D. Using Proposition 4.1.1, it is safe to assume that the support of D is in $E(G)$. By setting multiplicities as in the proof of Proposition 4.1.2, we can set this up to be an average stretch problem. Recall that a procedure to α-probabilistically embed a multigraph G into a distribution over spanning trees of G can be derandomized (see Section 2.6) to obtain a procedure that outputs a single spanning tree T of G such that $\text{av\_str}_T(G) \leq \alpha$. This implies a procedure to find a single spanning tree T of G such that $\text{com-cost}_T(G) \leq \alpha \cdot \text{com-cost}(G)$ and hence an α-approximation to the MCST instance. □

4.2 Undirected Formulation

4.2.1 LP Formulation

In [15] the following linear program formulation was suggested for MCST.

$$\min \quad \sum_{i,j \in V} D_{ij} \sum_{e \in E} w(e) f_{ij}(e) \tag{4.3}$$

$$\text{subject to} \quad x(E[S]) \leq |S| - 1 \qquad \forall S \subset V \tag{4.4}$$

$$x(E) = |V| - 1 \tag{4.5}$$

$$x(e) \geq 0 \qquad \forall e \in E \tag{4.6}$$

$$f_{ij}(\delta^+(v)) = f_{ij}(\delta^-(v)) \qquad \forall i, j \in V,\ \forall v \neq i, j \tag{4.7}$$

$$f_{ij}(\delta^+(i)) = f_{ij}(\delta^-(i)) + 1 \qquad \forall i, j \in V \tag{4.8}$$

$$f_{ij}(\delta^+(j)) = f_{ij}(\delta^-(j)) - 1 \qquad \forall i, j \in V \tag{4.9}$$

$$f_{ij}(e) \leq x(e) \qquad \forall i, j \in V,\ \forall e \in E \tag{4.10}$$

$$f_{ij}(e) \geq 0 \qquad \forall i, j \in V,\ \forall e \in E \tag{4.11}$$

The constraints (4.4) to (4.6) define x to be in the spanning tree polytope of G (Edmonds [10]). Recall that the vertices of the spanning tree polytope are the $\{0,1\}$ incidence vectors of spanning trees, i.e. the spanning tree polytope is $\mathrm{conv}\big(\chi^{E(T)} : T \text{ is a spanning tree}\big)$. A vector in the spanning tree polytope of G is thus a convex combination of such incidence vectors. The constraints (4.7) to (4.11) set up f_{ij} to be a unit flow from i to j obeying the edge capacities x. Thus, the support of each unit ij flow is a convex combination of spanning trees. For a solution (x, f_{ij}), the load on an edge e is $\sum_{i,j \in V} D_{ij} f_{ij}(e)$.

The LP can be compactly written as follows where $\mathrm{ST}(G)$ is the spanning tree polytope of G.

$$\min \quad \sum_{i,j \in V} \sum_{e \in E} D_{ij} w(e) f_{ij}(e) \tag{4.12}$$

$$\text{subject to} \quad x \in \mathrm{ST}(G) \tag{4.13}$$

$$f_{ij} \text{ is a unit flow from } i \text{ to } j \tag{4.14}$$

$$f_{ij}(e) \leq x(e) \qquad \forall i, j \in V,\ \forall e \in E \tag{4.15}$$

4.2.2 $\Omega(\log n)$ Integrality Gap

As noted in [15], the above LP formulation suffers a $\Omega(\log n)$ integrality gap and thus can not be used to improve the existing $O(\log n \cdot \log\log n)$ approximation guarantee for MCST. We now give a formal proof of this integrality gap.

Proposition 4.2.1. *The linear program formulation for MCST given by Equations (4.3) to (4.11) has a $\Omega(\log n)$ integrality gap for the infinite family of unit-weight diamond graphs $\{G_k\}_{k \in \mathbb{N}}$.*

Recall from Section 1.4.2 that the k-th diamond graph, $G_k = (V_k, E_k)$, can be recursively obtained from four copies of the $(k-1)$-th diamond graph, G_{k-1}. Recall also that we have $|V_k| = n_k = 4n_{k-1} - 4 \implies n_k = \frac{4^{k+1}+8}{6}$ and $|E_k| = m_k = 4m_{k-1} \implies m_k = 4^k$. Note that $m_k = \Theta(n_k)$ and $k = \Theta(\log n_k)$.

Define $\mathrm{MCST}(G, D)$ to be the optimal value of an MCST instance given by a graph G and a demand matrix D. Similarly, define $\mathrm{LP}_{\mathrm{ST}}(G, D)$ to be the optimal value of the linear program formulation given by Equations (4.3) to (4.11).

Lemma 4.2.2. *For the infinite family of unit-weight diamond graphs $\{G_k\}_{k\in\mathbb{N}^+}$ with the demand matrix D given by*

$$D_{ij} = \begin{cases} 1 & \text{if } ij \in E_k \\ 0 & \text{otherwise} \end{cases}$$

we have $MCST(G_k, D) = \Omega(n_k \log n_k)$.

Proof: For the family of unit-weight diamond graphs $\{G_k\}_{k\in\mathbb{N}}$, Gupta et al. [18] showed that for any $k \geq 1$ and any tree metric $d_T \geq d_{G_k}$ on V_k we have:

$$\sum_{e=uv\in E_k} d_T(u, v) = \Omega(k) \cdot \sum_{e=uv\in E_k} d_{G_k}(u, v) = \Omega(k) \cdot 4^k \tag{4.16}$$

(cf. Theorem 5.6 in [18] or Theorem 2.2.1 in this thesis)

Let T^* be the optimal solution to the given MCST instance. Then, for any $k \geq 1$, we have

$$\mathrm{MCST}(G_k, D) = \sum_{i,j\in V_k} D_{ij} d_{T^*}(i, j) = \sum_{e=ij\in E_k} d_{T^*}(i, j).$$

and so by (4.16) we get

$$\mathrm{MCST}(G_k, D) = \sum_{e=uv \in E_k} d_{T^*}(u,v) = \Omega(k) \cdot 4^k = \Omega(n_k \log n_k).$$

□

Lemma 4.2.3. *For an infinite family of unit-weight diamond graphs* $\{G_k\}_{k\in\mathbb{N}^+}$ *with the demand matrix given by*

$$D_{ij} = \begin{cases} 1 & \text{if } ij \in E_k \\ 0 & \text{otherwise,} \end{cases}$$

we have $LP_{ST}(G_k, D) = O(n_k)$.

Before giving the proof of Lemma 4.2.3, we prove that the uniform vector is in the spanning tree polytope of the diamond graph.

Claim 4.2.4. *The uniform vector* $x_k = \frac{n_k - 1}{m_k}\mathbf{1} \in ST(G_k)$, *for* $k \geq 0$.

Proof: We first give a recursive technique to find spanning trees of a diamond graph. We will then generalize it for the uniform vector to inductively prove the claim. Let $\{G^i_{k-1}\}^4_{i=1}$ be the four copies of the $(k-1)$-st diamond graph in G_k with terminals $\{s^i_{k-1}\}^4_{i=1}$ and $\{t^i_{k-1}\}^4_{i=1}$. We find four forests $\{T^i_{k-1}\}^4_{i=1}$, where $T^i_{k-1} \subset G^i_{k-1}$, such that $T_k = \bigcup^4_{i=1} T^i_{k-1}$ is a spanning tree of G_k. $\{T^i_{k-1}\}^3_{i=1}$ are spanning trees of the three copies $\{G^i_{k-1}\}^3_{i=1}$ and are obtained recursively. As for the fourth forest T^4_{k-1}, note that there is already a $s^4_{k-1}t^4_{k-1}$-path in $T_k \setminus T^4_{k-1}$. For simplicity, we may view the fourth copy as having an edge $s^4_{k-1}t^4_{k-1}$. We say that such a graph is "*wired*" and denote it by $\bar{G}^4_{k-1}$. Let $\bar{T}^4_{k-1}$ be a spanning tree of $\bar{G}^4_{k-1}$ such that $s^4_{k-1}t^4_{k-1} \in E(\bar{T}^4_{k-1})$. We also call such a spanning tree "wired". Set $T^4_{k-1} = \bar{T}^4_{k-1} - \{s^4_{k-1}t^4_{k-1}\}$. Clearly

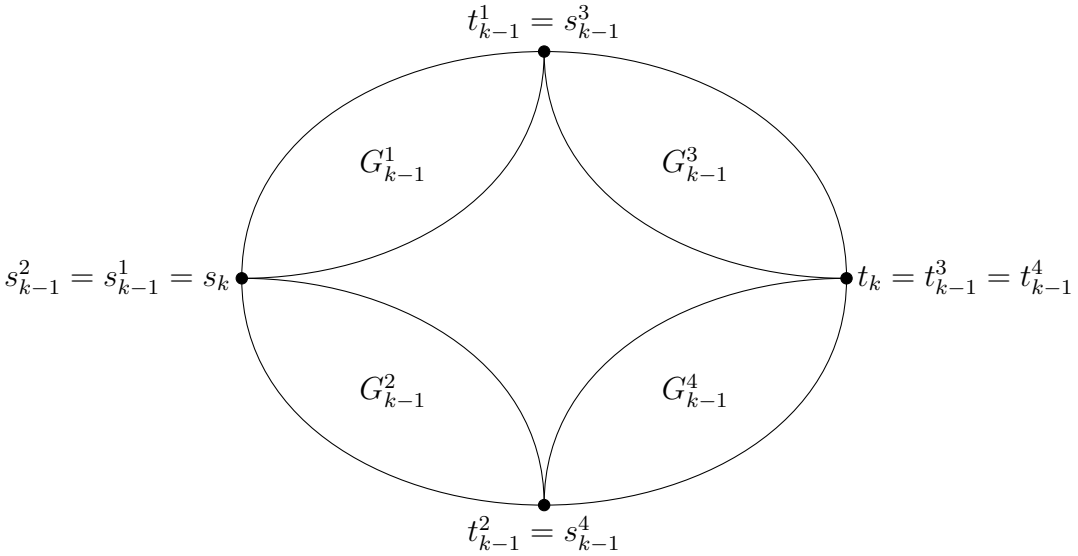


Figure 4–1: The Diamond Graph G_k with the four copies of G_{k-1}.

T^4_{k-1} is a forest in G^4_{k-1}. It is easy to see that $T_k = \bigcup_{i=1}^4 T^i_{k-1}$ is indeed a spanning tree of G_k. To complete the procedure we need a recursive technique to find a "wired" spanning tree $\bar{T}_k$ of the wired diamond graph $\bar{G}_k$. WLOG we fix the ordering of the four copies of G_{k-1}, as shown in Figure 4–1, by assuming that $s_k = s^1_{k-1} = s^2_{k-1}$ and $t_k = t^3_{k-1} = t^4_{k-1}$. Similar to before, we find four forests $\{T^i_{k-1}\}_{i=1}^4$, where $T^i_{k-1} \subset G^i_{k-1}$, such that $\bar{T}_k = \{s_k t_k\} \cup \bigcup_{i=1}^4 T^i_{k-1}$ is a "wired" spanning tree of $\bar{G}_k$. Now, T^1_{k-1} and T^2_{k-1} are the spanning trees of G^1_{k-1} and G^2_{k-1} respectively and can be obtained recursively by using the procedure described above. We recursively find "wired" spanning trees $\bar{T}^3_{k-1}$ and $\bar{T}^4_{k-1}$ of $\bar{G}^3_{k-1}$ and $\bar{G}^4_{k-1}$ respectively. Set $T^3_{k-1} = \bar{T}^3_{k-1} - \{s^3_{k-1}t^3_{k-1}\}$ and $T^4_{k-1} = \bar{T}^4_{k-1} - \{s^4_{k-1}t^4_{k-1}\}$. Clearly $\bar{T}_k = \{s_k t_k\} \cup \bigcup_{i=1}^4 T^i_{k-1}$ is a "wired" spanning tree of $\bar{G}_k$.

We now consider the uniform vectors. Let $\bar{x}_k$ be the "uniform" vector on $E(\bar{G}_k)$ as follows.

$$\bar{x}_k(e) = \begin{cases} 1 & e = s_k t_k \\ \frac{n_k - 2}{m_k} & \forall e \in E(\bar{G}_k) \setminus \{s_k t_k\}, \end{cases}$$

i.e. it is uniform on $E(\bar{G}_k) \setminus \{s_k t_k\}$ (the original edges from G_k). Similar to the procedures above, we may take uniform vectors for $\{G_{k-1}^i\}_{i=1}^3$ (resp $\{G_{k-1}^i\}_{i=1}^2$) and $\bar{G}_{k-1}^4$ (resp $\{\bar{G}_{k-1}^i\}_{i=3}^4$) to get the vector $x_k(e)$ for G_k (resp $\bar{x}_k(e)$ for $\bar{G}_k$). However, the vector $x_k(e)$ (resp $\bar{x}_k(e)$) will not be unifom. Instead, we symmetrize over the choice of "wired" copies by selecting each copy with probability 1/4. For the "wired" case we select G_{k-1}^1 and G_{k-1}^2 to be "wired" with probability 1/2 and we select G_{k-1}^3 and G_{k-1}^4 to be "wired" with probability 1/2. Indeed, by using $n_k = 4n_{k-1} - 4$ and $m_k = 4m_{k-1}$, for any edge $e \in E(G_k)$, we have

$$\begin{aligned} x_k(e) &= \frac{n_k - 1}{m_k} \\ &= \frac{4n_{k-1} - 5}{4m_{k-1}} \\ &= \frac{3(n_{k-1} - 1) + n_{k-1} - 2}{4m_{k-1}} \\ &= \frac{3}{4}\frac{n_{k-1} - 1}{m_{k-1}} + \frac{1}{4}\frac{n_{k-1} - 2}{m_{k-1}}. \end{aligned}$$

Similarly, for $\bar{x}_k(e)$ for $e \in E(\bar{G}_k) \setminus \{s_k t_k\}$, we have

$$\begin{aligned} \bar{x}_k(e) &= \frac{n_k - 2}{m_k} \\ &= \frac{4n_{k-1} - 6}{4m_{k-1}} \\ &= \frac{2(n_{k-1} - 1) + 2(n_{k-1} - 2)}{4m_{k-1}} \end{aligned}$$

$$= \frac{1}{2}\frac{n_{k-1}-1}{m_{k-1}} + \frac{1}{2}\frac{n_{k-1}-2}{m_{k-1}}.$$

Since $x_k(e) \geq 0$ and $x_k(E(G_k)) = \frac{n_k-1}{m_k}|E(G_k)| = n_k - 1 = |V(G_k)| - 1$, Equations 4.5 and 4.6 are satisfied by construction of x_k. Similarly, $\bar{x}_k(e) \geq 0$ and $\bar{x}_k(E(\bar{G}_k)) = 1 + \frac{n_k-2}{m_k}|E(G_k)| = n_k - 1 = |V(\bar{G}_k)| - 1$. So, what is left to show is that $x_k(E_k[S]) \leq |S| - 1$, for any set $S \subset V(G_k)$, and $\bar{x}_k(\bar{E}_k[S]) \leq |S| - 1$, for any set $S \subset V(\bar{G}_k)$. Assume, by induction, that

$$x_{k-1}(E_{k-1}[S]) = \frac{n_{k-1}-1}{m_{k-1}}|E_{k-1}[S]| \leq |S| - 1$$

and

$$\bar{x}_{k-1}(E_{k-1}[S]) = \frac{n_{k-1}-2}{m_{k-1}}|E_{k-1}[S]| \leq |S| - 2.$$

Note[2] that $\bar{x}_{k-1}(\bar{E}_{k-1}[S]) \leq 1 + \bar{x}_{k-1}(E_{k-1}[S]) \leq 1 + |S| - 2 = |S| - 1$. For any set of nodes $S \subset V(G_k)$, let $S_i = S \cap V(G^i_{k-1})$. That is, S_i is the part of S in the i-th copy of G_{k-1}. Note that $|E_k[S]| = \sum_{i=1}^4 |E_{k-1}[S_i]|$ while $\sum_{i=1}^4 |S_i| \leq |S| + 4$. We thus have

$$\begin{aligned}
x_k(E_k[S]) &= (\frac{3}{4}\frac{n_{k-1}-1}{m_{k-1}} + \frac{1}{4}\frac{n_{k-1}-2}{m_{k-1}})|E_{k-1}[S]| \\
&= \frac{3}{4}\frac{n_{k-1}-1}{m_{k-1}}\sum_{i=1}^4 |E_{k-1}[S_i]| + \frac{1}{4}\frac{n_{k-1}-2}{m_{k-1}}\sum_{i=1}^4 |E_{k-1}[S_i]| \\
&= \frac{3}{4}\sum_{i=1}^4 \frac{n_{k-1}-1}{m_{k-1}}|E_{k-1}[S_i]| + \frac{1}{4}\sum_{i=1}^4 \frac{n_{k-1}-2}{m_{k-1}}|E_{k-1}[S_i]|
\end{aligned}$$

[2]We use $\bar{x}_k(E_k[S])$ to denote $\bar{x}_k(\bar{E}_k[S] \setminus \{s_k t_k\})$.

$$\leq \frac{3}{4}\sum_{i=1}^{4}(|S_i|-1)+\frac{1}{4}\sum_{i=1}^{4}(|S_i|-2)$$
$$\leq \frac{3}{4}|S|+\frac{1}{4}(|S|-4)=|S|-1.$$

Similarly,

$$\begin{aligned}\bar{x}_k(E_k[S]) &= (\frac{1}{2}\frac{n_{k-1}-1}{m_{k-1}}+\frac{1}{2}\frac{n_{k-1}-2}{m_{k-1}})|E_{k-1}[S]| \\ &= \frac{1}{2}\frac{n_{k-1}-1}{m_{k-1}}\sum_{i=1}^{4}|E_{k-1}[S_i]|+\frac{1}{2}\frac{n_{k-1}-2}{m_{k-1}}\sum_{i=1}^{4}|E_{k-1}[S_i]| \\ &= \frac{1}{2}\sum_{i=1}^{4}\frac{n_{k-1}-1}{m_{k-1}}|E_{k-1}[S_i]|+\frac{1}{2}\sum_{i=1}^{4}\frac{n_{k-1}-2}{m_{k-1}}|E_{k-1}[S_i]| \\ &\leq \frac{1}{2}\sum_{i=1}^{4}(|S_i|-1)+\frac{1}{2}\sum_{i=1}^{4}(|S_i|-2) \\ &\leq \frac{1}{2}|S|+\frac{1}{2}(|S|-4)=|S|-2.\end{aligned}$$

Now, the base case for $k=0$ is trivial for both (one may verify it easily for $k=1$ as well since $k=0$ creates a parallel edge for the "wired" case) and thus we are done by induction. □

Proof of Lemma 4.2.3: Consider the uniform vector $x_k=\frac{n_k-1}{m_k}\mathbf{1}$. By Claim 4.2.4, $x_k\in ST(G_k)$. Now, for any $k\geq 1$ and any edge $e\in E(G_k)$, we have

$$x_k(e)=\frac{n_k-1}{m_k}=\frac{\frac{4^{k+1}+8}{6}-1}{4^k}=\frac{2}{3}+\frac{1}{3}(\frac{1}{4})^k\geq\frac{2}{3}.$$

Now, by the construction of D, we only have to consider the demand D_{ij} if $ij\in E_k$. By the construction of G_k, both i and j must be in a single copy of G_1

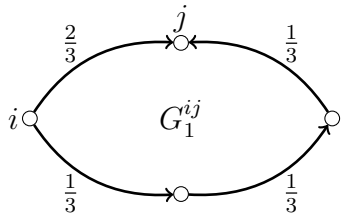


Figure 4–2: The f_{ij} solution for an edge ij. The edge ij lies in a single copy of G_1

(recall that G_1 is a 4-cycle), say G_1^{ij}. We define a valid unit flow f_{ij} as follows.

$$f_{ij}(e) = \begin{cases} \frac{2}{3} & \text{if } e = ij \\ \frac{1}{3} & \text{if } e \in E(G_1^{ij}) \setminus \{ij\} \\ 0 & \text{otherwise.} \end{cases}$$

It is easy to see that f_{ij} induces a unit flow from i to j (see Figure 4–2). Note also that $f_{ij}(e) \leq \frac{2}{3} \leq x_k(e)$, for any $e \in E_k$, and thus it obeys capacities x_k. Since $|E(G_1^{ij})| = 4$, we have

$$\begin{aligned} \mathrm{LP}_{\mathrm{ST}}(G_k, D) &\leq \sum_{i,j \in V_k} \sum_{e \in E_k} D_{ij} w(e) f_{ij}(e) \\ &= \sum_{ij \in E_k} \sum_{e \in E_k} f_{ij}(e) \\ &= \sum_{ij \in E_k} (\frac{2}{3} + \sum_{e \in E(G_1^{ij}) \setminus \{ij\}} \frac{1}{3}) \\ &= \sum_{ij \in E_k} (\frac{2}{3} + 3\frac{1}{3}) \\ &= \frac{5}{3} m_k = O(n_k). \end{aligned}$$

This completes the proof. □

We are now ready to prove Proposition 4.2.1.

Proof of Proposition 4.2.1: Consider the infinite family of MCST instances on unit-weight diamond graphs $\{(G_k, D)\}_{k\in\mathbb{N}^+}$ as defined in Lemmas 4.2.2 and 4.2.3. By the same two lemmas, the integrality gap is given by

$$\frac{\mathrm{MCST}(G_k, D)}{\mathrm{LP}_{\mathrm{ST}}(G_k, D)} = \frac{\Omega(n_k \log n_k)}{O(n_k)} = \Omega(\log n_k),$$

which completes the proof. □

4.3 Directed Formulation

4.3.1 Weak Directed Formulation

We propose to strengthen this LP formulation by using the directed setting and using spanning arborescences intead of spanning trees. Let $r \in V$ be a chosen root for the spanning arborescences. For each demand pair ij, create a dummy node v_{ij} connected to each $u \in V(G)$ with a "fake" edge (u, v_{ij}). Let $F(G)$ be the set of these "fake" edges for the graph G. Our first linear programming formulation is as follows, where we replace each uv edge with a (u, v) edge and a (v, u) edge. We call this the Weak Directed Formulation (we will strengthen this further later in the section).

$$\min \quad \sum_{i,j\in V} D_{ij} \sum_{e\in E} w(e)(h^i_{ij}(e) + h^j_{ij}(e)) \tag{4.17}$$

$$\text{subject to} \quad x(\delta^+(v)) = 1 \qquad \forall v \in V \setminus \{r\} \tag{4.18}$$

$$x(\delta^-(R)) \geq 1 \qquad \forall r\text{-cut } \delta^-(R) \subset V \tag{4.19}$$

$$x(e) \geq 0 \qquad \forall e \in E \cup F \tag{4.20}$$

$$x(\delta^-(v_{ij})) = 1 \qquad \forall i, j \in V \tag{4.21}$$

$$h^{\ell}_{ij}(\delta^+(v)) = h^{\ell}_{ij}(\delta^-(v)) \qquad \forall \ell \in \{i,j\},\ \forall i,j \in V,\ \forall v \neq \ell \tag{4.22}$$

$$h^{\ell}_{ij}(\delta^+(\ell)) = h^{\ell}_{ij}(\delta^-(\ell)) + 1 \qquad \forall \ell \in \{i,j\},\ \forall i,j \in V \tag{4.23}$$

$$h^{\ell}_{ij}(\delta^-(v_{ij})) = 1 \qquad \forall \ell \in \{i,j\},\ \forall i,j \in V \tag{4.24}$$

$$h^{\ell}_{ij}(e) \leq x(e) \qquad \forall \ell \in \{i,j\},\ \forall i,j \in V,\ \forall e \in E \cup F \tag{4.25}$$

$$h^{\ell}_{ij}(e) \geq 0 \qquad \forall \ell \in \{i,j\},\ \forall i,j \in V,\ \forall e \in E \cup F \tag{4.26}$$

Similar to the Undirected Formulation, the constraints (4.18) to (4.20) define x to be in the spanning r-arborescence polytope of G. Recall that the vertices of the spanning r-arborescence polytope are incidence vectors of spanning r-arborescences of G. A vector in the spanning r-arborescence polytope of G is thus a convex combination of incidence vectors of spanning r-arborescences of G. The constraints (4.22) to (4.26) set up h^i_{ij} to be a unit flow from i to v_{ij} using x capacities. Note that for each ij pair there will be two unit flows, namely h^i_{ij} and h^j_{ij}. The constraint (4.21) ensures that these flows use the same "fake" edges for the flow to the dummy node v_{ij}. The union of these two flows gives paths in the original graph G for the ij pair to communicate. If x is integral, then each ij pair will transfer one unit of flow as follows. i sends one unit of flow to the least common ancestor of i and j ($LCA(i,j)$) and then to v_{ij}. This is the flow h^i_{ij}. j sends one unit of flow to $LCA(i,j)$ and then to v_{ij}. This is the flow h^j_{ij}. The load on an edge e is $\sum_{i,j \in V} D_{ij}(h^i_{ij}(e) + h^j_{ij}(e))$.

The LP can be compactly written as follows where r-ARB(G) is the spanning r-arborescence polytope of G.

$$\min \quad \sum_{i,j \in V} \sum_{e \in E} D_{ij} w(e)(h^i_{ij}(e) + h^j_{ij}(e)) \tag{4.27}$$

$$\text{subject to} \quad x(E) \in r\text{-ARB}(G) \tag{4.28}$$

$$x(\delta^-(v_{ij})) = 1 \qquad \forall i, j \in V \tag{4.29}$$

$$h^i_{ij} \text{ is a unit flow from } i \text{ to } v_{ij} \tag{4.30}$$

$$h^i_{ij}(e) \leq x(e) \qquad \forall i, j \in V,\ \forall e \in E \cup F \tag{4.31}$$

One may reformulate the constraint $x \in r$-ARB(G) (constraints (4.18) to (4.20)) as below to avoid the exponentially many r-cut constraints.

$$x(\delta^+(v)) = 1 \qquad \forall v \in V \setminus \{r\} \tag{4.32}$$

$$x(e) \geq 0 \qquad \forall e \in E \tag{4.33}$$

$$f^i(\delta^+(i)) = 1 \qquad \forall i \in V \setminus \{r\} \tag{4.34}$$

$$f^i(\delta^-(r)) = 1 \qquad \forall i \in V \setminus \{r\} \tag{4.35}$$

$$f^i(\delta^+(v)) = f^i(\delta^-(v)) \qquad \forall v \in V \setminus \{i, r\}, \forall i \in V \setminus \{r\} \tag{4.36}$$

$$f^i(e) \leq x(e) \qquad \forall e \in E, \forall i \in V \setminus \{r\} \tag{4.37}$$

Each f^i vector is a unit flow from i to r obeying x capacties on the edges.

Claim 4.3.1. *The polytope defined by (4.32) to (4.37) equals the spanning r-arborescence polytope (defined by (4.18) to (4.20)).*

Proof: Note that we only have to show that constraints (4.34) to (4.37) are equivalent to the constraint (4.19). Any feasible point in the spanning r-arborescence

polytope allows a unit flow from each $i \in V \setminus \{r\}$ to r. Similarly, if there is a unit flow from each $i \in V \setminus \{r\}$ to r, then (4.19) is immediately satisfied. $\square$

4.3.2 Strong Directed Formulation

We strengthen the Weak Directed Formulation even further by adding restrictions to the h^i_{ij} (resp h^j_{ij}) flows. Instead of just obeying the x-capacities, they must actually obey capacities induced by the flow vectors f^i (resp f^j) on the real edges. In some sense, node i "commits" to f^i which it must use for all of its communication h_{ij} vectors. The Strong Directed Formulation for MCST is then as follows.

$$\min \quad \sum_{i,j \in V} D_{ij} \sum_{e \in E} w(e)(h^i_{ij}(e) + h^j_{ij}(e)) \tag{4.38}$$

subject to

$$x(\delta^+(v)) = 1 \qquad \forall v \in V \setminus \{r\} \tag{4.39}$$

$$x(e) \geq 0 \qquad \forall e \in E \tag{4.40}$$

$$x(\delta^-(v_{ij})) = 1 \qquad \forall i, j \in V \tag{4.41}$$

$$f^i(\delta^+(i)) = 1 \qquad \forall i \in V \setminus \{r\} \tag{4.42}$$

$$f^i(\delta^-(r)) = 1 \qquad \forall i \in V \setminus \{r\} \tag{4.43}$$

$$f^i(\delta^+(v)) = f^i(\delta^-(v)) \qquad \forall v \in V \setminus \{i, r\}, \forall i \in V \setminus \{r\} \tag{4.44}$$

$$f^i(e) \leq x(e) \qquad \forall e \in E, \forall i \in V \setminus \{r\} \tag{4.45}$$

$$h^\ell_{ij}(\delta^+(v)) = h^\ell_{ij}(\delta^-(v)) \qquad \forall \ell \in \{i, j\},\ \forall i, j \in V,\ \forall v \neq \ell \tag{4.46}$$

$$h^\ell_{ij}(\delta^+(\ell)) = h^\ell_{ij}(\delta^-(\ell)) + 1 \qquad \forall \ell \in \{i, j\},\ \forall i, j \in V \tag{4.47}$$

$$h^\ell_{ij}(\delta^-(v_{ij})) = 1 \qquad \forall \ell \in \{i, j\},\ \forall i, j \in V \tag{4.48}$$

$$h^{\ell}_{ij}(e) \leq f^{\ell}(e) \qquad \forall \ell \in \{i,j\},\ \forall i,j \in V,\ \forall e \in E \tag{4.49}$$

$$h^{\ell}_{ij}(e) \leq x(e) \qquad \forall \ell \in \{i,j\},\ \forall i,j \in V,\ \forall e \in F \tag{4.50}$$

$$h^{\ell}_{ij}(e) \geq 0 \qquad \forall \ell \in \{i,j\},\ \forall i,j \in V,\ \forall e \in E \cup F \tag{4.51}$$

We were recently made aware that [26] also considered a Linear Programming based formulation for approximating MCST. Our Strong Directed Formulation is different from their approach in two key aspects. First, the x vector in our formulation is actually in the spanning arborescence polytope. Second, for a pair of nodes $i, j \in V$ the unit flows h^i_{ij} and h^j_{ij} must abide by the capacities induced by the flow vectors f^i and f^j respectively.

4.3.3 Semi-Strong Directed Formulation

We relax the Strong Directed Formulation given by (4.38) to (4.51). For each pair of nodes $i, j \in V(G)$, we replace the two unit flows, h^i_{ij} and h^j_{ij} obeying f^i and f^j capacities respectively, with a single unit flow h_{ij} from i to j obeying capacities as follows. For an edge $e = (u, v)$, we use $\bar{e}$ to refer to the edge (v, u). We have $h_{ij}(e) \leq f^i(e) + f^j(\bar{e})$. The Semi-Strong Directed Formulation is then as follows.

$$\min \qquad \sum_{i,j \in V} D_{ij} \sum_{e \in E} w(e)(h_{ij}(e)) \tag{4.52}$$

$$\text{subject to} \qquad x(\delta^+(v)) = 1 \qquad \forall v \in V \setminus \{r\} \tag{4.53}$$

$$x(e) \geq 0 \qquad \forall e \in E \tag{4.54}$$

$$x(\delta^-(v_{ij})) = 1 \qquad \forall i, j \in V \tag{4.55}$$

$$f^i(\delta^+(i)) = 1 \qquad \forall i \in V \setminus \{r\} \tag{4.56}$$

$$f^i(\delta^-(r)) = 1 \qquad \forall i \in V \setminus \{r\} \tag{4.57}$$

$$f^i(\delta^+(v)) = f^i(\delta^-(v)) \qquad \forall v \in V \setminus \{i, r\}, \forall i \in V \setminus \{r\} \tag{4.58}$$

$$f^i(e) \leq x(e) \qquad \forall e \in E, \forall i \in V \setminus \{r\} \tag{4.59}$$

$$h_{ij}(\delta^+(v)) = h_{ij}(\delta^-(v)) \qquad \forall i, j \in V,\ \forall v \neq i \tag{4.60}$$

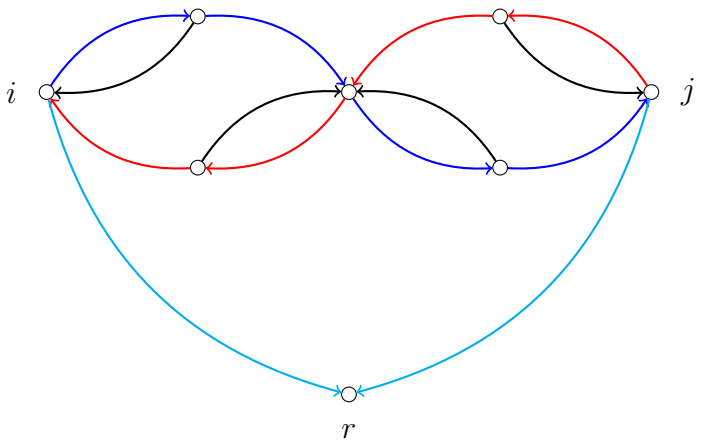
$$h_{ij}(\delta^+(i)) = h_{ij}(\delta^-(i)) + 1 \qquad \forall i, j \in V \tag{4.61}$$

$$h_{ij}(\delta^+(j)) = h_{ij}(\delta^-(j) - 1 \qquad \forall i, j \in V \tag{4.62}$$

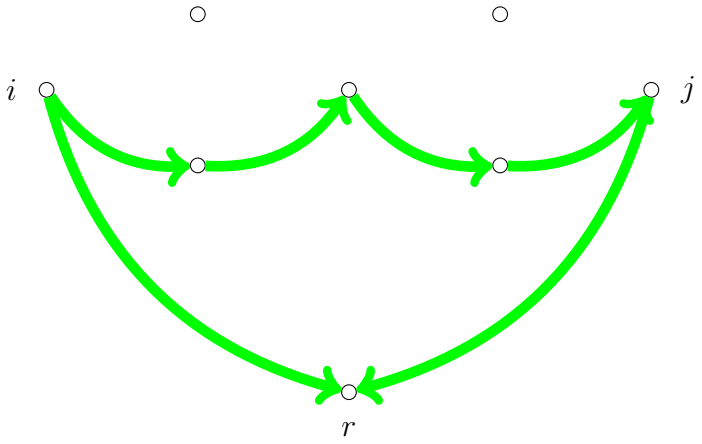
$$h_{ij}(e) \leq f^i(e) + f^j(\bar{e}) \qquad \forall i, j \in V,\ \forall e \in E \tag{4.63}$$

$$h_{ij}(e) \geq 0 \qquad \forall i, j \in V,\ \forall e \in E \tag{4.64}$$

Clearly, every h^i_{ij} and h^j_{ij} flows in the Strong Directed Formulation correspond to a h_{ij} flow in the relaxed formulation. However, the converse may not be true, i.e. not every h_{ij} flow in the Semi-Strong Directed Formulation can be converted into h^i_{ij} and h^j_{ij} flows in the Strong Directed Formulation. This can be seen by looking at a (x, f^i) solution given by Figure 4–3(a). Each edge e in the figure has $x(e) = 0.5$. For every cyan edge e, we have $f^i(e) = 0.5$ and $f^j(e) = 0.5$. For every blue edge e, we have $f^i(e) = 0.5$ and $f^j(e) = 0$. For every red edge e, we have $f^i(e) = 0$ and $f^j(e) = 0.5$. For every other edge e, we have $f^i(e) = 0$ and $f^j(e) = 0$. Figure 4–3(b) gives a solution for h_{ij} flow in the Semi-Strong Directed Formulation which does not correspond to any h^i_{ij} and h^j_{ij} flows in the Strong Directed Formulation.



(a)



(b)

Figure 4–3: A solution to the Semi-Strong Directed Formulation that does not correspond to any solution of the Strong Directed Formulation.

For an MCST instance given by a graph G and a demand matrix D, we denote the value of the Strong Directed Formulation by $\mathrm{LP}_{\mathrm{SDF}}(G, D)$ and the value of the Semi-Strong Directed Formulation by $\mathrm{LP}_{\mathrm{SSDF}}(G, D)$. We have the following relationship between the values of the three LP formulations suggested in this chapter.

$$\mathrm{LP}_{\mathrm{ST}}(G, D) \leq \mathrm{LP}_{\mathrm{SSDF}}(G, D) \leq \mathrm{LP}_{\mathrm{SDF}}(G, D) \leq \mathrm{MCST}(G, D).$$

CHAPTER 5
Conclusion

The cornerstone problem in this thesis was the Minimum Communication cost Spanning Tree problem. We have surveyed the results in metric embeddings and described how they relate to approximations to MCST. In order to improve the approximation ratio, we presented linear programming based formulations for MCST. The Undirected Formulation was suggested previously by [15]. Unfortunately, it suffers from an $\Omega(\log n)$ integrality gap. We have proposed stronger formulations for MCST using a directed setting. At the time of writing of this thesis, we were still working on understanding the integrality gap of these formulations.

These formulations also provide a framework to obtain computational results for MCST. Initial computer simulations suggest that the Strong Directed Formulation has a long running time for the diamond graph G_k with $k > 3$. However, very few attempts have been made to optimize the implementation. No simulations have yet been attempted for the Semi-Strong Directed Fromulation. Since the number of variables is much smaller than the Strong Directed Formulation, we expect the running times to be shorter.

References

[1] Ittai Abraham, Yair Bartal, and Ofer Neiman. Nearly tight low stretch spanning trees. In *Foundations of Computer Science, 2008. FOCS'08. IEEE 49th Annual IEEE Symposium on*, pages 781–790. IEEE, 2008.

[2] Ittai Abraham and Ofer Neiman. Using petal-decompositions to build a low stretch spanning tree. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pages 395–406. ACM, 2012.

[3] Noga Alon, Richard M Karp, David Peleg, and Douglas West. A graph-theoretic game and its application to the k-server problem. *SIAM Journal on Computing*, 24(1):78–100, 1995.

[4] Noga Alon and Joel H Spencer. *The Probabilistic Method*. John Wiley, New York, 3 edition, 2008.

[5] Baruch Awerbuch. Complexity of network synchronization. *Journal of the ACM (JACM)*, 32(4):804–823, 1985.

[6] Yair Bartal. Probabilistic approximation of metric spaces and its algorithmic applications. In *Foundations of Computer Science, 1996. Proceedings., 37th Annual Symposium on*, pages 184–193. IEEE, 1996.

[7] Yair Bartal. On approximating arbitrary metrices by tree metrics. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 161–168. ACM, 1998.

[8] Joshua Batson, Daniel A Spielman, Nikhil Srivastava, and Shang-Hua Teng. Spectral sparsification of graphs: theory and algorithms. *Communications of the ACM*, 56(8):87–94, 2013.

[9] Chandra Chekuri, Anupam Gupta, Ilan Newman, Yuri Rabinovich, and Alistair Sinclair. Embedding k-outerplanar graphs into l1. *SIAM Journal on Discrete Mathematics*, 20(1):119–136, 2006.

[10] Jack Edmonds. Matroids and the greedy algorithm. *Mathematical programming*, 1(1):127–136, 1971.

[11] Michael Elkin, Yuval Emek, Daniel A Spielman, and Shang-Hua Teng. Lower-stretch spanning trees. *SIAM Journal on Computing*, 38(2):608–628, 2008.

[12] Yuval Emek. k-outerplanar graphs, planar duality, and low stretch spanning trees. *Algorithmica*, 61(1):141–160, 2011.

[13] Yuval Emek and David Peleg. A tight upper bound on the probabilistic embedding of series-parallel graphs. *SIAM Journal on Discrete Mathematics*, 23(4):1827–1841, 2009.

[14] Jittat Fakcharoenphol, Satish Rao, and Kunal Talwar. A tight bound on approximating arbitrary metrics by tree metrics. *Journal of Computer and System Sciences*, 69:485–497, 2004.

[15] Alexandre Frechette, Neil Olver, and F. Bruce Shepherd. Attempt at a linear program formulation for the minimum communication cost spanning tree problem. Unpublished manuscript, August 2013.

[16] Ralph E Gomory and Tien Chung Hu. Multi-terminal network flows. *Journal of the Society for Industrial & Applied Mathematics*, 9(4):551–570, 1961.

[17] Anupam Gupta. Steiner points in tree metrics don't (really) help. In *Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms*, pages 220–227. Society for Industrial and Applied Mathematics, 2001.

[18] Anupam Gupta, Ilan Newman, Yuri Rabinovich, and Alistair Sinclair. Cuts, trees and l_1-embeddings of graphs. *Combinatorica*, 24(2):233–269, 2004.

[19] Te C Hu. Optimum communication spanning trees. *SIAM Journal on Computing*, 3(3):188–195, 1974.

[20] David S Johnson, Jan Karel Lenstra, and AHG Kan. The complexity of the network design problem. *Networks*, 8(4):279–285, 1978.

[21] Richard M Karp. A $2k$-competitive algorithm for the circle. Manuscript, August 1989.

[22] David Peleg. Approximating minimum communication spanning trees. In *SIROCCO'97, 4th International Colloquium on Structural Information & Communication Complexity, Monte Verita, Ascona, Switzerland, July 24-26, 1997*, pages 1–11, 1997.

[23] Yuri Rabinovich and Ran Raz. Lower bounds on the distortion of embedding finite metric spaces in graphs. *Discrete & Computational Geometry*, 19(1):79–94, 1998.

[24] Harald Räcke, Chintan Shah, and Hanjo Täubig. Computing cut-based hierarchical decompositions in almost linear time. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '14, pages 227–238. SIAM, 2014.

[25] Prabhakar Raghavan. Probabilistic construction of deterministic algorithms: Approximating packing integer programs*. *Journal of Computer and System Sciences*, 37:130–143, 1988.

[26] Eilon Reshef. Approximating minimum communication cost spanning trees and related problems. Master's thesis, Weizmann Institute of Science, Rehovot, Israel, 1999.

[27] Kunal Talwar. *Metric Methods in Approximation Algorithms*. PhD thesis, Berkeley, CA, USA, 2004. AAI3165575.

[28] Bang Ye Wu, Giuseppe Lancia, Vineet Bafna, Kun-Mao Chao, R Ravi, and Chuan Yi Tang. A polynomial-time approximation scheme for minimum routing cost spanning trees. *SIAM Journal on Computing*, 29(3):761–778, 2000.