# A Computational Study of a Geometric Embedding of Minimum Multiway Cut

by

David Shin

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrial Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

[June 2006]
May 2006

Author ..................................................................
Department of Electrical Engineering and Computer Science
May 16, 2006

[illegible]

Certified by .......[illegible].......
David R. Karger
Associate Professor
Thesis Supervisor

[illegible]

Accepted by .......[illegible].......
Arthur C. Smith
Chairman, Department Committee on Graduate Students

# A Computational Study of a Geometric Embedding of Minimum Multiway Cut

by

David Shin

Submitted to the Department of Electrical Engineering and Computer Science
on May 16, 2006, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrial Engineering and Computer Science

## Abstract

In the minimum multiway cut problem, the goal is to find a minimum cost set of edges whose removal disconnects a certain set of $k$ distinguished vertices in a graph. The problem is MAX-SNP hard for $k \geq 3$. Călinescu, Karloff, and Rabani gave a geometric relaxation of the problem and a rounding scheme, to produce an approximation algorithm that has a performance guarantee of $3/2 - 1/k$. In a subsequent study, Karger, Klein, Stein, Thorup, and Young discovered improved rounding schemes via computation experiments for various values of $k$, yielding approximation algorithms with improved performance guarantees. Their rounding scheme for $k = 3$ is provably optimal (i.e., its performance guarantee is equal to the integrality gap of the relaxation), but their rounding schemes for $k > 3$ seemed unlikely to be optimal.

In the present work, we improve these rounding schemes for small values of $k > 3$, yielding improved approximation algorithms. These improvements were discovered by applying an improved analysis to the same set of computational experiments used by Karger et al. We also present computer-aided proofs of improved lower bounds on the integrality gap for various values of $k > 3$. For the $k = 4$ case, for instance, our work demonstrates a lower and upper bound of 1.1052 and 1.1494, respectively, improving upon the previously best known bounds of 1.0909 and 1.1539. Finally, we present additional computational experiments that may shed some light on the nature of the optimal rounding scheme for the $k = 4$ case.

Thesis Supervisor: David R. Karger
Title: Associate Professor

## Acknowledgments

I would like to thank my parents and my sister Sarah. Their support over the past years has been vital to me.

Dina Katabi and Dan Stratila allowed me to use their machines for my computational experiments. I would like to thank them for their generosity.

Finally, I must thank David Karger for the many useful ideas and insights into this problem. I am grateful for his enthusiasm and advice.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

In the minimum multiway cut problem, one is given a weighted graph and a size-$k$ subset of the vertices called the terminals. The goal is to find a minimum cut whose removal disconnects the terminals from each other. The problem is MAX-SNP hard for $k \geq 3$, so we are interested in approximation algorithms.

We are particularly interested in a geometric relaxation of the problem formulated by Călinescu, Karloff, and Rabani. Their approach uses the common graph optimization technique of embedding a graph into a geometric space. In order to round this relaxed solution to a cut of the graph, one partitions the geometric space, as a partitioning of the geometric space naturally induces a cut in the graph. By partitioning the geometric space randomly according to a carefully chosen distribution, one can use the properties of the embedding to prove bounds on the expected value of the induced cut, thus generating an approximation algorithm.

Specifically, we have sought to investigate the integrality gap of this geometric relaxation. This value represents the best provable approximation factor based on Călinescu et al.'s relaxation. The question has been resolved for the $k = 3$, but remains open for $k > 3$.

In the present work, we establish tighter lower and upper bounds on the exact value of the integrality gap for various values of $k$. Furthermore, we analyze some previous conjectures concerning the yet-to-be-discovered optimal rounding schemes for $k > 3$, while presenting some conjectures of our own.

## 1.1 Problem Definition

We begin with some preliminary definitions and notation. Given a graph $G = (V, E)$, and a nonnegative weight function $w : E \rightarrow \mathbb{R}$, a *cut* is a subset $C \subseteq E$, and the *cost of* $C$ is given by $w(C) = \sum_{e \in C} w(e)$. When we are additionally given a distinguished subset $T \subseteq V$, we define a *$T$-cut* to be a cut whose removal disconnects every pair of vertices of $T$. In this context, the vertices of $T$ are referred to as *terminals.* If $k = |T|$, we also call this type of cut a *$k$-way cut* of $G$.

In the *minimum multiway cut problem*, we are given as input $G$, $w$, and $T$, and seek to find a $k$-way cut of minimum cost. We notate the cost of the minimum $k$-way cut of a graph $G$ by $w_0(G)$. Note that the minimum multiway cut problem is a natural generalization of the classical $(s, t)$-cut problem.

The minimum multiway cut problem arises in several applications. One may formulate the scheduling of jobs on multiple processors in distributed computing systems as a minimum multiway cut problem [11]. In computer vision, maximum *a posteriori* estimates of a certain useful class of Markov Random Fields can be obtained by solving minimum multiway cut [2]. Other applications include partitioning files among the nodes of a network and partitioning the elements of a circuit during the design of electronic chips into the subcircuits that are placed on different chips [4].

## 1.2 Prior Work

The minimum multiway cut problem was first studied by Dahlaus, Johnson, Papadimitriou, Seymour, and Yannakakis [4]. They showed that the problem was MAX-SNP hard for $k \geq 3$. Additionally, they gave a simple approximation algorithm: for each terminal, $t$, use the traditional minimum $(s, t)$-cut algorithm [5] as a subroutine to find the minimum cut separating $t$ from the other terminals, and take the union of the $k - 1$ smallest such cuts. One can prove that this algorithm achieves an approximation factor of $2 - 2/k$.

Călinescu, Karloff, and Rabani [3] then gave a novel geometric relaxation for the problem. Their idea was to embed the graph into the $k$-simplex, which is a $(k-1)$-dimensional polytope in $\mathbb{R}^k$ with $k$ vertices[1]. This embedding is done in such a way that the $k$ terminals map to the $k$ simplex vertices. Then, the simplex is partitioned into $k$ regions, so that each vertex of the simplex lies in a different region. This partitioning, referred to as a *$k$-way cut of the simplex*, induces a $k$-way cut in the original graph in a natural way: an edge of the original graph is in the cut if its endpoints map to different regions of the partitioned simplex. The manner in which this partitioning is chosen (i.e., a probability distribution over $k$-way cuts of the simplex) is referred to as a *cutting scheme*. Călinescu et al. specify a simple cutting scheme and use the properties of the embedding to show that the resulting approximation algorithm achieves a performance guarantee of $3/2 - 1/k$.

Subsequently, Karger, Klein, Stein, Thorup, and Young [8] extended this work by developing cutting schemes for various values of $k$ that yield improved approximation ratios. These cutting schemes were found through computational experiments involving probability distributions over a class of partitions of the simplex that they refer to as *side-parallel cuts*, abbreviated *sparcs*. They also gave a precise geometric criterion for proving the optimality of a cutting scheme. They then used this criterion to prove the optimality of their cutting scheme for the case $k = 3$, thus demonstrating the limits of the Călinescu et al. embedding for $k = 3$. The question of the optimal cutting scheme for $k > 3$, however, remained open.

Mosk-Aoyama, in his Master's thesis, considered alternatives to sparcs for the $k = 4$ case. He performed computational experiments involving a new class of partitions of the simplex, which he refers to as *pair-isolating cuts*. The results of his computational experiments led him to conjecture that no optimal rounding scheme for $k > 3$ can be defined purely in terms of sparcs.

[1]As illustrative examples, the 3-simplex is an equilateral triangle, and the 4-simplex is a regular tetrahedron. We define the $k$-simplex exactly in Chapter 2.

## 1.3 Our Results

Our goal is to further understand the geometric relaxation of Călinescu et al. Specifically, we aim to establish tighter lower and upper bounds on the *integrality gap* of the relaxation for various values of $k$. The integrality gap represents the best approximation ratio one can prove using an analysis that bounds the optimum cut only by the value of the relaxation.

We were successful on both fronts. We found stronger upper bounds by applying a more careful analysis to the computational experiments of Karger et al. We found stronger lower bounds through an original set of computational experiments. For $k = 4$, we obtain bounds of 1.1052 and 1.1494, improving upon the corresponding Karger et al. bounds of 1.0909 and 1.1539. In the process, we devise a branch-and-bound heuristic algorithm to solve the minimum multiway cut problem exactly.

We also perform additional computational experiments for the $k = 4$ case to explore the possible role of pair-isolating cuts in optimal cutting schemes. Our conjecture, on the basis of these additional experiments, is that an optimal cutting scheme can be defined without using pair-isolating cuts. A bolder conjecture is that an optimal cutting scheme can be defined purely in terms of sparcs.

## 1.4 Presentation Overview

In Chapter 2, we describe in detail the geometric embedding of Călinescu et al., and the works of Karger et al. and Mosk-Aoyama. Chapter 3 discusses how we were able to tighten the analysis underlying Karger et al.'s computational experiments. The performance guarantees given at the end of Chapter 3 represent the best known values to date. In Chapter 4, we present our branch-and-bound heuristic algorithm to solve minimum multiway cut exactly, and show how we were able to use this algorithm to establish new lower bounds on the integrality gap of the relaxation. These bounds, given at the end of Chapter 4, also represent the best known values to date.

Chapter 5 is of a more speculative nature. In it, we discuss some additional computational experiments we have performed to investigate pair-isolating cuts. We believe these experiments to suggest that pair-isolating cuts are not a necessary component of the optimal cutting scheme. Concluding remarks and natural directions for future research are given in Chapter 6.

# Chapter 2

# Background

Our present work draws heavily upon the previous works of Călinescu et al., Karger et al., and Mosk-Aoyama. In this chapter, we introduce the key ideas and results of these previous works.

## 2.1 The Geometric Relaxation of Călinescu, Karloff, and Rabani

The basis of this work is the geometric relaxation of the minimum multiway cut problem formulated by Călinescu et al. In order to lay the framework for this relaxation, we begin with some terminology and notation.

### 2.1.1 The embedding

The geometric space we consider is the *k-simplex* $\Delta = \{x \in \mathbb{R}^k | \sum_i x_i = 1 \wedge x \geq 0\}$. Note that $\Delta$ is a polytope of $\mathbb{R}^k$ with $k$ vertices. The $i$-th vertex is the point identified by the $i$-th coordinate vector $e^i$, which has coordinates $(e^i)_i = 1$ and $(e^i)_j = 0$ for all $j \neq i$. To avoid ambiguity, we refer to the vertices of the graph, the elements of $V$, as *nodes*, and reserve the word *vertices* for the vertices of the simplex.

We measure distance in the simplex by using the $L_1$-norm divided by 2. Thus, if $x$ and $y$ are points of the simplex, we have $d(x, y) = \frac{1}{2}\sum_i |x_i - y_i|$. If $s$ is the line segment in the simplex with endpoints $x$ and $y$, we define the length of $s$ to be $|s| = d(x, y)$. The factor of $\frac{1}{2}$ is present to scale the distance between simplex vertices to 1.

An *embedding* of a graph $G = (V, E)$ into $\Delta$ is a mapping $\alpha : V \longrightarrow \Delta$. For any $e = (u, v) \in E$, we let $\alpha(e)$ denote the line segment connecting $\alpha(u)$ and $\alpha(v)$. We define the *volume* of $\alpha$ by

$$\text{vol}(\alpha) = \sum_{e \in E} w(e)|\alpha(e)|.$$

Consider an embedding that maps each node to a vertex of the simplex, with the $i$th terminal mapping to the $i$th simplex vertex. Then each embedded edge has length 0 if its endpoints embed to the same vertex and 1 otherwise. Every such embedding naturally induces a $k$-way cut of the graph. In this context, the set of edges in the cut are the edges whose endpoints map to different vertices of the simplex. Thus, the volume of the embedding is equal to the value of the induced cut.

These observations motivate the following integer program formulation (IP) of the minimum multiway cut problem:

$$\begin{aligned} &\min \text{vol}(\beta) \ s.t. \\ &\beta \text{ is an embedding into } \Delta \\ &\beta(u) \in \{e^1, e^2, \ldots, e^k\} \qquad \forall u \in V \\ &\beta(t) = e^t \qquad \forall t \in T \end{aligned} \tag{IP}$$

Here, we are assuming that the terminals are labeled $1, 2, \ldots, k$.

### 2.1.2 The relaxation

The integer problem is, of course, intractable, as it is equivalent to the original MAX-SNP hard minimum multiway cut problem. However, by removing the integer constraints, we obtain a tractable linear programming relaxation (LP):

$$\begin{aligned} &\min \operatorname{vol}(\alpha) \ s.t. \\ &\alpha \text{ is an embedding into } \Delta \\ &\alpha(t) = e^t \qquad\qquad \forall t \in T \end{aligned} \tag{LP}$$

Expressing LP as a linear program is not trivial, as the objective function involves distances between embedded points. Distances are of the form $d(x, y) = \sum_i |x_i - y_i|$, and absolute values are not permitted in linear programs. In order to get around this difficulty, we instead loosen such constraints to read $d(x, y) \geq \sum_i |x_i - y_i|$. This is permissible, since minimizing the objective function will force equality at every such inequality. We then replace the given inequality with the following equivalent set of inequalities, which do not use absolute values:

$$\begin{aligned} d(x,y) &\geq \sum_i d_i(x,y) \\ d_i(x,y) &\geq x_i - y_i \\ d_i(x,y) &\geq y_i - x_i. \end{aligned}$$

.

A feasible solution to LP is a mapping $\alpha$ from $V$ to arbitrary points of $\Delta$. We denote the optimal solution by $\alpha^*$. To convert a feasible solution $\alpha$ into a feasible solution $\beta$ of IP, we partition the simplex into $k$ regions, one per vertex of the simplex:

$$\Delta = \bigcup_{i=1}^{k} R_i \quad s.t. \quad e^i \in R_i \;\; \forall i$$

We call such a partitioning a *k-way cut* of $\Delta$. Our converted solution, $\beta$, is such that $\beta(u) = e^i$ iff $\alpha(u) \in R_i$.

With a $k$-way cut of $\Delta$ and a feasible solution $\alpha$ to LP, we can generate a feasible solution to IP, and thus a $k$-way cut of the original graph. Specifically, each region of the simplex generated by the $k$-way cut corresponds to a connected component of the graph after the cut edges are removed. This means that we add edge $(u, v)$ to the cut iff $\alpha(u)$ and $\alpha(v)$ lie in different regions of the simplex.

### 2.1.3 Rounding schemes

In order to turn this framework into an algorithm that finds $k$-way cuts of the graph, we must specify how to choose the $k$-way cut of the simplex. One option is to use *randomized rounding* by specifying a probability distribution over $k$-way cut of $\Delta$. We call such a probability distribution a *cutting scheme.*

Note that a cutting scheme specifies how to "round" the solution of a fractional program (LP) to a solution of an integer program (IP). For this reason, we use the term *rounding scheme* interchangably with "partitioning scheme".

The rounding scheme used by Călinescu et al. is independent of the input graph. It has the property that the $k$-way cut of the graph induced by the random $k$-way cut has an expected cost at most $3/2-1/k$ times the volume of the computed embedding. The induced algorithm thus has an approximation ratio of $3/2 - 1/k$.

### 2.1.4 Integrality gap

The approach of embedding-relaxation-rounding outlined thus far is a formulaic one, used throughout the literature to approximately solve many NP-hard graph optimization problems. In every such application of this methodology, the question of the value of the *integrality gap* of the relaxation arises.

Formally, in our framework, the integrality gap of the relaxation is the supremum, over all weighted graphs $G$, of the ratio of the minimum cost of any $k$-way cut of $G$ divided by the volume of the minimum volume embedding of $G$. In other words, it is equal to the worst case ratio between the optimal value of the integer program IP and the optimal value of the linear program LP.

This value is significant as it represents the best approximation ratio one can prove using an analysis that bounds the value of the rounded solution only by the value of the relaxed solution.

## 2.2 A Study of the Relaxation by Karger, Klein, Stein, Thorup, and Young

The work of Karger et al. provides additional insight into the relaxation, as well as further results about its integrality gap. Among other things, they show that the problems of determining the integrality gap and finding a corresponding optimal rounding scheme can be expressed purely as a geometric question. Thus, the integrality gap and approximation ratio can be obtained by studying the simplex itself, without considering particular input graphs or embeddings.

### 2.2.1 Density

Let us say that a line segment $s$ in $\Delta$ is *cut* by a $k$-way cut $C$ if some region boundary of $C$ intersects $s$. Let $\phi_k(C, s)$ denote the number of times that $C$ cuts $s$.

Given a probability distribution $P$ over $k$-way cuts of $\Delta$ (i.e., a cutting scheme), and a line segment, $s$, in $\Delta$, let the *density of $P$ on $s$* be given by

$$\tau_k(P, s) = \mathrm{E}_{C \in P}\left[\frac{\phi_k(C, s)}{|s|}\right].$$

We then define the *maximum density of $P$*, $\tau_k(P)$, by

$$\tau_k(P) = \sup_s \tau_k(P, s).$$

The supremum is taken over all segments $s$ with endpoints in $\Delta$.

The relevance of $\tau_k(P)$ is the following:

**Lemma 1 (Karger et al.).** *A cutting scheme $P$ yields a randomized approximation algorithm with approximation ratio at most $\tau_k(P)$.*

*Proof.* Let $\alpha$ be the minimum cost embedding. Consider an edge $e$ in $G$. This edge maps to a segment $\alpha(e)$, which is cut in expectation at most $\tau_k(P, \alpha(e)) \cdot |\alpha(e)|$ times. By the Markov inequality, this upper bounds the probability that the edge is cut.

Thus, the expected cost of the induced $k$-way cut is at most

$$\begin{aligned}\sum_e (\tau_k(P, \alpha(e)) \cdot |\alpha(e)|)\, w(e) &\leq \tau_k(P) \sum_e |\alpha(e)| w(e) \\ &= \tau_k(P)\mathrm{vol}(G).\end{aligned}$$

Since LP is a relaxation of IP, the minimum volume lower bounds the minimum $k$-way cut. The lemma follows. □

Lemma 1 suggests that we should search for cutting schemes $P$ with the property that $\tau_k(P)$ is small. Thus, we define the *minimum maximum density*, $\tau_k^*$, by

$$\tau_k^* = \inf_P \tau_k(P),$$

and frame our goal as finding the value of $\tau_k^*$. Here, the infimum is taken over all probability distributions over $k$-way cuts. In fact, the work of Karger et al. indicates something stronger:

**Theorem 2 (Karger et al.).** *There exists a cutting scheme whose maximum density equals the integrality gap. Thus, $\tau_k^*$ equals the integrality gap.*

The proof is highly technical and so we omit it here. The theorem is significant, as it implies that the value of $\tau_k^*$ can be resolved exactly by the following twofold strategy:

1. Find a family[1] of cutting schemes $\{P_i\}$ such that

$$\lim_{i \to \infty} \tau_k(P_i) = \tau_k^*.$$

2. Find a family of graphs, $\{G_i\}$ such that

$$\lim_{i \to \infty} \left( \frac{w_0(G_i)}{\mathrm{vol}(\alpha^*(G_i))} \right) = \tau_k^*.$$

[1]Of course, if a *single* cutting scheme $P$ could be found such that $\tau_k(P) = \tau_k^*$, there would be no need to find an entire family. However, there is no guarantee that such a cutting scheme exists. Similar remarks apply for the family of graphs in 2.

Recall that $C_0(G)$ denotes the minimum $k$-way cut of $G$, and that $\alpha^*(G)$ denotes an optimal embedding of $G$.

That the family of cutting schemes exists is implied directly by the definition of $\tau_k^*$ (with or without the theorem), but the existence of the family of graphs relies on the theorem. If the theorem were not true, it is unclear how one would identify a particular value as $\tau_k^*$.

### 2.2.2 Maximum density segments

The work of Karger at al. demonstrates that the key property of a cutting scheme is the maximum density the cutting scheme has on any line segment in the simplex. Implicit in the work of Călinescu et al. is the observation that it is only necessary to consider infinitesimal segments in certain orientations. This is captured by the following lemmas, which we will not prove.

**Lemma 3.** *There is always a line segment of infinitesimal length that achieves the maximum density.*

We say that a line segment is $i, j$*-aligned* if it is parallel to the edge whose endpoints are vertices $i$ and $j$ of the simplex. We say it is *aligned* if it is $i, j$-aligned for some $i, j$.

**Lemma 4.** *There is always an aligned line segment that achieves the maximum density.*

These lemmas simplify the task of computing $\tau_k(P)$.

### 2.2.3 Some important linear programs

In this section, we introduce some linear programs that will be referred to extensively throughout this work.

**The upper bounding linear program**

Karger et al. noted that the problem of finding an optimal cutting scheme could be formulated as an infinite-dimensional linear program, with one variable for each $k$-way cut of the simplex, and one constraint for every infinitesimal aligned segment. The variables represent probabilities, thus inducing a cutting scheme, and each constraint bounds the density of the induced cutting scheme on the corresponding segment. The objective function minimizes the density bound. The solution to the linear program is an assigment of probabilities to the various $k$-way cuts, or a cutting scheme.

The linear program is as follows:

$$\begin{aligned} &\min \tau \\ &\sum_C p_C = 1 \\ \sup_s &\sum_C p_C \cdot \frac{\phi_k(C,s)}{|s|} \leq \tau \end{aligned} \qquad \text{(PRI-}\infty\text{)}$$

Here, the sums are taken over the set of all $k$-way cuts $C$ of the simplex.

Actually, a much more precise formulation is required, as the set of all $k$-way cuts is uncountable, making the sums in the constraints meaningless. The variables should rather take the form of a measure defined on some $\sigma$-field of subsets of the set of all $k$-way cuts of the simplex, and the constraints should rather take the form of Lebesgue integrals[2]. Nevertheless, we present this (incorrectly formulated) infinite linear program for sake of simplicity. Throughout this work, we will only use discretized versions of PRI-$\infty$, for which an appropriate transformation from measures and Lebesgue integrals to discrete probability distributions and sums can be made.

In particular, the approach taken by Karger et al. was to partition a subset of the uncountably many $k$-way cuts into a finite collection of subsets, $\mathcal{Q}$. They assigned a probability distribution over each $Q \in \mathcal{Q}$, so that $\mathcal{Q}$, together with some probabilities $\{p_Q\}_{Q \in \mathcal{Q}}$ summing to 1, induced a probability distribution over the set of all $k$-way cuts: namely, choose a $Q \in \mathcal{Q}$ with probability $p_Q$, and then choose a cut from $Q$ according to the probability distribution assigned to it. We call each $Q \in \mathcal{Q}$ a *sub-distribution*.

[2]The reader is directed to [1] for a technical treatment of these measure theory concepts.

Furthermore, Karger et al. partitioned the simplex into a finite collection of *cells*, $\mathcal{W}$, and observed that the supremum of PRI-$\infty$ could be taken over all segments $s$ wholly contained in some cell $W \in \mathcal{W}$ (by Lemma 3). For any $X \subseteq \Delta$, let $S(X)$ denote the set of line segments whose endpoints lie in $X$. The linear program PRI-$\infty$ then becomes a legitimate one:

$$\begin{aligned} &\min \tau \\ &\sum_Q p_Q = 1 \\ \sup_{s \in S(W)} &\sum_Q p_Q \cdot \tau_k(Q, s) \leq \tau \qquad \forall W \in \mathcal{W} \end{aligned} \tag{PRI-$\infty_2$}$$

The significance of this linear program is summarized by the following theorem.

**Theorem 5.** *The objective value of PRI-$\infty_2$ is equal to the maximum density of the cutting scheme specified by its variables.*

Still, the suprema in the constraints represent an infinite number of constraints, making the linear program impractical, though legitimate. To get around this, let

$$\theta(Q, W) = \sup_{s \in S(W)} \tau_k(Q, s),$$

and note that for any subset $W \subseteq \Delta$,

$$\sup_{s \in S(W)} \sum_Q p_Q \cdot \tau_k(Q, s) \leq \sum_Q p_Q \cdot \sup_{s \in S(W)} \tau_k(Q, s) = \sum_Q p_Q \cdot \theta(Q, W). \tag{2.1}$$

By computing an upper bound, $\psi(Q, W)$, on $\theta(Q, W)$, for each $Q \in \mathcal{Q}$ and $W \in \mathcal{W}$, we may use (2.1) to arrive at a finite linear program:

$$\begin{aligned} &\min \tau \\ &\sum_Q p_Q = 1 \\ &\sum_Q p_Q \cdot \psi(Q, W) \leq \tau \qquad \forall W \in \mathcal{W} \end{aligned} \tag{LP-UB}$$

The significance of this linear program is summarized by the following theorem.

**Theorem 6.** *The objective value of LP-UB is an upper bound of the maximum density of the cutting scheme specified by its variables.*

*Remark* 2.2.1. The two factors that can cause an optimal solution of LP-UB to be non-optimal for PRI-$\infty_2$ are:

- The slack lost in the application of (2.1), and
- The gap between $\psi(Q, W)$ and $\theta(Q, W)$.

### The lower bounding linear program

In a similar fashion, the problem of finding a worst-case graph can be formulated as an infinite-dimensional linear program. By "worst-case graph", we mean a graph $G$ for which the ratio $\frac{w_0(G)}{\mathrm{vol}(\alpha^*(G))}$ is maximal. Since the given ratio can only decrease by replacing $\alpha^*$ with a non-optimal embedding $\alpha$, we can in fact formulate an infinite-dimensional linear program to find a worst-case *embedded* graph: that is, a graph $G$ together with an embedding $\alpha$ for which the ratio $\frac{w_0(G)}{\mathrm{vol}(\alpha(G))}$ is maximal. Note that an embedded graph is simply a set of embedded nodes, $\alpha(1), \alpha(2), \ldots, \alpha(k), \ldots, \alpha(n)$, where $\alpha(i)$ is set equal to the $i$-th vertex of the simplex for $1 \leq i \leq k$, together with an assignment of edge weights for each pair of embedded nodes[3]. For full generality, our set of embedded nodes can be taken to equal the entire simplex, in which case the embedded graph can be fully expressed by an assignment of edge weights to simplex segments.

Thus, this infinite linear program has a variable for every segment of the simplex, and a constraint for every $k$-way cut of the simplex. The variables represent edge weights, and the constraints bound the cost of the minimum $k$-way cut of the graph. One further constraint is used to set the volume of the embedding to 1, so that the quantity to be maximized is simply the cost of the minimum $k$-way cut of the graph. This can be done because the ratio $\frac{w_0(G)}{\mathrm{vol}(\alpha(G))}$ is invariant under a scaling of the edge

[3]We may use a set (as opposed to a multiset) of embedded nodes, since if two nodes embed to the same point of the simplex, they will never be separated by a cut, allowing us to consider instead the graph formed by merging the two nodes into one.

weights. The objective function maximizes the cost of the minimum $k$-way cut of the graph.

The linear program is as follows:

$$\begin{aligned} &\max \lambda \\ &\sum_s w_s \cdot |s| = 1 \\ &\inf_C \sum_{s \text{ cut by } C} w_s \geq \lambda \end{aligned} \tag{DUAL-$\infty$}$$

The infimum is taken over all $k$-way cuts $C$ of the simplex, and the sums are taken over all segments of the simplex. As before, the sums are meaningless as they are taken over uncountable sets, making this linear program formulation erroneous. Again, we point out that we will only use discretized versions of DUAL-$\infty$, for which an appropriate transformation from measures and Lebesgue integrals to discrete weight assignments and sums can be made.

In particular, one natural approach is to consider a fixed embedding of a finite graph. We can do this by selecting a finite subset of points in the the simplex, $V \subseteq \Delta$, and by defining a graph, $G(V)$, whose vertices are the points of $V$, and whose edges are the elements of $S(V)$. We let cuts($V$) denote the set of $k$-way cuts of $G(V)$. The infimum of DUAL-$\infty$ then becomes a minimum over a finite numbers of cuts. The linear program DUAL-$\infty$ then becomes

$$\begin{aligned} &\max \lambda \\ &\sum_{s \in S(V)} w_s \cdot |s| = 1 \\ &\sum_{s \in C} w_s \geq \lambda \qquad \forall C \in \text{cuts}(V) \end{aligned} \tag{LP-LB}$$

We call this linear program LP-LB, since the resultant value of $\lambda$ represents a lower bound on $\tau_k^*$, with the resultant $\{w_s\}$ specifying an embedded graph that achieves that value. Unlike the linear program LP-UB, however, it is not clear that this linear program can be of much practical computational use, as the number of constraints is exponential in the size of the embedded graph. Karger et al. were able to bypass

$$\begin{tikzcd}
\boxed{\text{PRI-}\infty} \arrow[r, leftrightarrow, "\text{dual}"] \arrow[d, "\text{discretize}"'] & \boxed{\text{DUAL-}\infty} \arrow[dd, "\text{discretize}"] \\
\boxed{\text{PRI-}\infty_2} \arrow[d, "\text{slack}"'] & \\
\boxed{\text{LP-UB}} & \boxed{\text{LP-LB}}
\end{tikzcd}$$

Figure 2-1: The relationship between the linear programs PRI-$\infty$, PRI-$\infty_2$, DUAL-$\infty$, LP-UB, and LP-LB.

this difficulty for the $k = 3$ case by exploiting planarity to replace the exponentially many constraints with a provably equivalent set of polynomially many constraints.

One can see that this DUAL-$\infty$ is actually the dual of PRI-$\infty$. This, along with strong duality, is in fact the key observation behind Theorem 2. The argument, however, requires some more rigor than what we have presented. Karger et al. take a more rigorous measure-theory based approach by considering finite discrete versions of both linear programs and by taking a limit as the discrete programs approach the given continuous ones.

The relationship between the linear programs PRI-$\infty$, PRI-$\infty_2$, DUAL-$\infty$, LP-UB, and LP-LB are summarized by Figure 2-1.

### 2.2.4 Side parallel cuts

For small values of $k$, Karger et al. were able to solve instances of LP-UB using the linear program software CPLEX. In this section, we describe a particular type of cut they used in their set up. They refer to this type of cut as a *side parallel cut*, or *sparc.*

Define $\Delta_{x_i=\rho} = \{x \in \Delta | x_i = \rho\}$ and $\Delta_{x_i \geq \rho} = \{x \in \Delta | x_i \geq \rho\}$ (similarly $\Delta_{x_i<\rho}$). Note that $\Delta_{x_i=\rho}$ is a hyperplane that runs through the simplex, parallel to the face opposite vertex $i$, at a distance $\rho$ from that face. We call such a hyperplane a *side parallel slice.* This hyperplane cuts the simplex into two regions: the *corner*, $\Delta_{x_i \geq \rho}$, and the *base*, $\Delta_{x_i<\rho}$.

A sparc is defined by Karger et al. as a $k$-way cut of the simplex (that is, a partition of $\Delta$ into regions $1, 2, \ldots, k$) resulting from the following procedure.

1. Choose a permutation $\sigma$ of the simplex vertices.
2. Process the vertices in the order specified by $\sigma$. For each vertex $i$, except the last, choose a slice distance $\rho_i \in [0, 1]$.
3. When vertex $i$ is processed, assign to region $i$ all the points in $\Delta_{x_i \geq \rho_i}$ (the corner) that have not already been assigned to a previous region. We say that vertex $i$ *captures* the points assigned to region $i$, and that it *cuts* a segment if it captures some of the points in the segment, but not the entire segment.
4. After the first $(k-1)$ vertices have been processed, assign the remaining unassigned points of the simplex to the final $k$-th region.

When $\sigma$ is chosen uniformly at random from the set of all permutations, and the slice distances are $\rho_1, \rho_2, \ldots, \rho_{k-1}$, we call the resulting sparc a $[\rho_1, \rho_2, \ldots, \rho_{k-1}]$-*sparc*.

### 2.2.5 Computational experiments

We can now describe the particular setup Karger et al. used for LP-UB. For their setup, they fixed an integer *discretization level* $d$, and used one sub-distribution for each element of $\{0, 1, \ldots, d-1\}^{k-1}$. The sub-distribution for $(q_1, q_2, \ldots, q_{k-1})$ is induced by choosing a value $\rho_i$ uniformly at random from $[q_i/d, (q_i+1)/d]$ for each $i$, and then by taking a $[\rho_1, \rho_2, \ldots, \rho_{k-1}]$-sparc. We refer to this specific sub-distribution as a *sparc range*.

For their cells, they used the regions of $\Delta$ formed by slicing the simplex along the hyperplanes $x_i = j/d$ for $1 \leq i \leq k$ and $1 \leq j \leq d-1$. Thus each cell takes the form

$$\{(x_1, \ldots, x_k) : w_i/d \leq x_i \leq (w_i+1)/d\}.$$

With this choice of cells and sub-distributions, a set of upper bounds $\psi(Q, W)$ could be computed. Section 3.1 discusses how exactly Karger et al. computed these bounds. In Section 3.2, we will show how we were able to compute stronger bounds.

For the $k = 3$ case, Karger et al. were also able to solve instances of the linear program LP-LB for appropriately chosen embedded graphs. The embedded graph they used was chosen by fixing an integer discretization level $d$ and by using all points in the simplex of the form $(a_1/d, a_2/d, a_3/d)$ with $a_1, a_2, a_3 \in \mathbb{Z}$ as embedded nodes. Rather than listing the exponentially many constraints, however, they were able to rely on max-flow/min-cut duality and the planarity of the simplex to devise an equivalent linear program with only polynomially many constraints. They could not use an analogous strategy for $k > 3$, and so were unable to obtain better lower bounds on $\tau_k^*$ for $k > 3$.

### 2.2.6 Results

For the $k = 3$ case, Karger et al. were able to observe convergent behavior for the solutions of both LP-UB and LP-LB as the discretization level increased. This led to the following discoveries:

1. A cutting scheme $P$ for which they could prove analytically that $\tau_3(P) = 12/11$.

2. A family of embedded graphs, $\{\alpha_i(G_i)\}$ such that

$$\lim_{i\to\infty}\left(\frac{w_0(G_i)}{\text{vol}(\alpha_i(G_i))}\right) = \frac{12}{11}.$$

These discoveries together imply that $\tau_3^* = 12/11$.

For greater values of $k$, solutions to LP-UB led to computer generated proofs of upper bounds for $\tau_k^*$. These values are summarized in Table 2.1. Their experiments revealed an interesting fact: in all cases, the optimum cut distribution made use of "corner cuts." That is, the output distribution had the following form: with some probability, place each slice at a single distance chosen uniformly between 0 and 1/3 from its terminal; otherwise, use a (joint) distribution that places each slice at distance greater than 1/3 from its terminal.

| $k$ | discretization level | corner cut probability | bound |
|---|---|---|---|
| 4 | 36 | .289 | 1.1539 |
| 5 | 18 | .314 | 1.2161 |
| 6 | 12 | .376 | 1.2714 |
| 7 | 9 | .397 | 1.3200 |

Table 2.1: The upper bounds on the integrality gap computed by Karger et al.

## 2.3 Additional Computational Experiments by Mosk-Aoyama

Although the optimal cutting scheme for $k = 3$ can be expressed as a distribution over sparcs, there is no guarantee that this will be true for general $k$. Mosk-Aoyama, in his Master's thesis, explored alternatives to sparcs for the $k = 4$ case. He considered cuts that first isolate two pairs of vertices from each other, then separate each pair of vertices. He refers to such cuts as *pair-isolating cuts.* All such cuts he considered began with a *pair-isolating slice*: a hyperplane that is parallel to the two edges of the simplex connecting the pairs of vertices that they isolate. Such a hyperplane is shown in Figure 2-2. He then considered two different alternatives for how to separate the pairs of vertices. One was to separate them via a hyperplane parallel to a side of the simplex, and the other was to separate them via a hyperplane perpendicular to an edge of the simplex. He calls the pair-isolating cut induced by using a pair-isolating cut in conjunction with one of these two types of vertex-separating slices a *pair-isolating side-parallel cut* (abbreviated pair-side cut) and a *pair-isolating edge-perpendicular cut* (abbreviated pair-edge cut), respectively.

For his computational experiments, Mosk-Aoyama had to formulate appropriate versions of LP-UB, which entailed defining sub-distributions $\{Q_i\}$ and cells $\{W_j\}$, and computing density bounds $\psi(Q_i, W_j)$. For this, he first fixes a discretization level $d \geq 1$. He then defines sub-distributions over pair-side cuts and pair-edge cuts which we will call *pair-side cut ranges* and *pair-edge cut ranges* (analogous to Karger et al.'s sparc ranges), and uses as his sub-distributions the set of all sparc ranges,

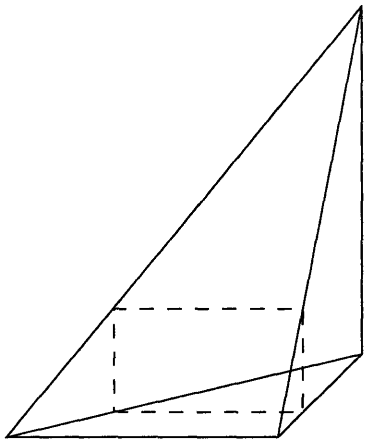

Figure 2-2: A pair-isolating slice. The intersection of the pair-isolating slice and the simplex is represented by the dashed region. This slice isolates the pair of vertices connected by the horizontal edge from the pair of vertices connected by the vertical edge.

pair-side cut ranges, and pair-edge cut ranges. For his cells, Mosk-Aoyama considers the possible orientations of all hyperplanar boundaries that can be generated by any cut chosen from his set of sub-distributions. He then performs $d$ equally spaced hyperplanar slices of the simplex along each one of these orientations, and takes the resultant regions of the simplex as his cells. Note that this can be seen as a natural generalization of the way in which Karger et al. define their cells with respect to their choice of sub-distributions.

Mosk-Aoyama's results indicate that at a fixed discretization level, the introduction of pair-side cuts and pair-edge cuts slightly improves the objective value of LP-UB. This led him to postulate that an optimal cutting scheme for $k = 4$ cannot be described as a probability distribution over sparcs. We will investigate this postulate in further detail in Chapter 5.

# Chapter 3

# Revisiting the Linear Program of Karger et al.

In Section 2.2.3, we described an infinite-dimensional optimization problem, PRI-$\infty$, that captures the problem of computing the minimum maximum density $\tau_k^*$. Karger et al. formulated the finite linear program LP-UB, whose objective value is provably an upper bound on $\tau_k^*$. In this chapter, we discuss how we were able to loosen some of the constraints of that linear program in order to obtain stronger upper bounds. These upper bounds are given in Table 3.1, near the end of this chapter.

## 3.1 The constraints of Karger et al.

Recall the discretized version of PRI-$\infty$ used by Karger et al. for their computational experiments:

$$\begin{aligned} &\min \tau \\ &\sum_Q p_Q = 1 \\ &\sum_Q p_Q \cdot \psi(Q, W) \le \tau \qquad \forall W \in \mathcal{W} \end{aligned} \tag{LP-UB}$$

Here, $\mathcal{W}$ is a collection of cells, and the sums are taken over $\mathcal{Q}$, a collection of sub-distributions. The values $\{p_Q\}$ specify a probability distribution over the sub-distributions. Each $\psi(Q, W)$ is an upper bound on the value $\theta(Q, W)$ defined by:

$$\theta(Q, W) = \sup_{s \in S(W)} \tau_k(Q, s).$$

Recall that $S(W)$ denotes the set of all segments whose endpoints lie in $W$. Let $S_{1,2}(W)$ denote the set of all $1,2$-aligned segments of $S(W)$, and define the quantity $\theta_{1,2}(Q, W)$ by

$$\theta_{1,2}(Q, W) = \sup_{s \in S_{1,2}(W)} \tau_k(Q, s).$$

Karger et al. noted that $\theta(Q, W) = \theta_{1,2}(Q, W)$. This fact follows from Lemma 4 and the fact that each sub-distribution $Q$ is symmetric with respect to all the vertices. We may thus focus our attention on $\theta_{1,2}(Q, W)$.

Karger et al. computed the upper bounds $\psi(Q, W)$ as follows. Note that each sparc range $Q$ can be itself thought of as a uniform probability distribution over $k!$ sub-distributions, one per permutation of $\{1, \ldots, k\}$. If we let $Q_\sigma$ be the sub-distribution corresponding to the permutation $\sigma$, we have that

$$\theta_{1,2}(Q, W) = \sup_{s \in S_{1,2}(W)} \frac{1}{k!} \sum_\sigma \tau_k(Q_\sigma, s). \tag{3.1}$$

Now, note that

$$\begin{aligned} \sup_{s \in S_{1,2}(W)} \sum_\sigma \tau_k(Q_\sigma, s) &\leq \sum_\sigma \sup_{s \in S_{1,2}(W)} \tau_k(Q_\sigma, s) \\ &= \sum_\sigma \theta_{1,2}(Q_\sigma, W). \end{aligned} \tag{3.2}$$

To compute a bound $\psi(Q, W)$, then, we can compute the values $\theta_{1,2}(Q_\sigma, W)$, and set

$$\psi(Q, W) = \frac{1}{k!} \sum_\sigma \theta_{1,2}(Q_\sigma, W). \tag{3.3}$$

To see that $\psi(Q,W)$ is an upper bound on $\theta_{1,2}(Q,W)$, note that

$$\begin{aligned}\theta_{1,2}(Q,W) &= \sup_{s\in S_{1,2}(W)} \frac{1}{k!}\sum_{\sigma}\tau_k(Q_\sigma,s)\\ &\le \sum_{\sigma}\frac{1}{k!}\cdot \sup_{s\in S_{1,2}(W)} \tau_k(Q_\sigma,s)\\ &= \frac{1}{k!}\sum_{\sigma}\theta_{1,2}(Q_\sigma,W)\\ &= \psi(Q,W).\end{aligned}$$

Here, we have made use of equations (3.1) and (3.2).

The values $\theta_{1,2}(Q_\sigma,W)$ can be computed as follows. The vertices are processed in the order specified by $\sigma$, and the $\ell$-th slice distance $\rho_\ell$ is applied to the $\ell$-th vertex in the permutation, $\sigma(\ell)$.

When applying the $\ell$-th slice distance, we have three distinct possibilities. If the $\ell$-th coordinate of the sparc range is different from that of the cell, then the $\ell$-th slice will not pass through the cell: depending on whether the coordinate is larger or smaller the slice will either capture the entire cell or none of the cell. If the $\ell$-th coordinates are the same, then the slice might pass through the cell. In this case, we can use the fact that the slice is uniformly distributed over a range.

A $1,2$-aligned segment can only be cut if the slices for vertex 1 or 2 go through the cell. If only one of the two slices goes through the cell, and no earlier slice goes through the cell, then the density of $Q_\sigma$ on the segment is exactly $d$ (the length of the segment divided by the width of the cell). If both slices go through the cell, the density is at most $2d$, implying that $\tau_k(Q_\sigma,W)\le 2d$, and thus that $\theta_{1,2}(Q_\sigma,W)\le 2d$.

In fact, we can see that $\theta_{1,2}(Q_\sigma,W)$ is exactly equal to $2d$ in the case when both slices go through the cell, and no earlier slice goes through the cell. It is not necessary to prove this in order to perform our goal of simply establishing an upper bound on $\theta_{1,2}(Q,W)$, but we include a proof for completeness:

**Lemma 7.** *When the slices for both vertex 1 and 2 go through the cell, and no earlier slice goes through the cell,*

$$\theta_{1,2}(Q_\sigma,W)=2d.$$

*Proof.* Assume without loss of generality that the slice for vertex 1 comes before the slice for vertex 2. Consider then an infinitesimally small segment $s$ that is arbitrarily close to the hyperplanes $x_1 = w_1$ and $x_2 = w_2$. Then, the probability that the first slice captures $s$ is negligible, implying that the density contributions of the two slices can be added independently, for a total density arbitrarily close to $d + d = 2d$. Thus, $\theta_{1,2}(Q_\sigma, W) > 2d - \epsilon$ for all $\epsilon > 0$, implying that $\theta_{1,2}(Q_\sigma, W) = 2d$. □

We can summarize our analysis by giving an explicit formula for $\theta_{1,2}(Q_\sigma, W)$. To do this, we first give an explicit formula for $\chi(Q_\sigma, W)$, the number of slices of $Q_\sigma$ that can cut a $1,2$-aligned segment contained in $W$. Let the coordinates of $Q$ be $(q_1, \dots, q_{k-1})$, and let $W$ be the cell given by

$$\{(x_1, \dots, x_k) : w_i/d \le x_i \le (w_i + 1)/d\}.$$

For any $i \in \{1, \dots, k\}$, let

$$f_i^\sigma(Q, W) = \begin{cases} 0 & \text{if } \exists m < \sigma^{-1}(i) \ : \ q_m < w_{\sigma(m)} \\ 1 & \text{otherwise.} \end{cases}$$

The value of $f_i^\sigma$ is 0 if some slice earlier than the $i$-th slice captures the cell $W$.

Define $\delta(m, n)$ to be 1 if $m = n$ and 0 otherwise.

Then, we have that

$$\chi(Q_\sigma, W) = \left(f_1^\sigma(Q, W)\delta(q_{\sigma^{-1}(1)}, w_1) + f_2^\sigma(Q, W)\delta(q_{\sigma^{-1}(2)}, w_2)\right).$$

Our explicit formula for $\theta_{1,2}(Q_\sigma, W)$ then reads:

$$\theta_{1,2}(Q_\sigma, W) = d \cdot \chi(Q_\sigma, W).$$

Combining this with equation (3.3), we obtain the density bound used by Karger et al.:

$$\psi(Q, W) = \frac{d}{k!} \sum_{\sigma} \chi(Q_\sigma, W). \tag{3.4}$$

## 3.2 Exact density bounds

The analysis of the previous section relies on the application of inequality (3.2). A certain amount of slack is lost in doing so, resulting in sub-optimal density bounds $\psi(Q, W)$. An intuitive explanation for this lost slack can be found in our proof of Lemma 7. In the proof, the segment $s$ for which $\tau_k(Q_\sigma, s)$ attains its maximum value is chosen *adaptively* with respect to the permutation $\sigma$. On the other hand, equality can clearly only take place in (3.2) if that segment is chosen *irrespective* of the permutation $\sigma$.

As a result, the computed objective value of LP-UB only represents an upper bound of the maximum density of the corresponding cutting scheme. In this section, we show how to calculate the values $\theta(Q, W)$ exactly. Furthermore, we prove that with these new density bounds, the computed objective value of LP-UB in fact represents the *exact* value of the maximum density of the corresponding cutting scheme.

We begin by defining an alternate type of sub-distribution, closely related to a sparc range. As with sparc ranges, we fix an integer discretization level $d$ and use one sub-distribution for each element of $\{0, 1, \ldots, d-1\}^{k-1}$. The sub-distribution for $(q_1, q_2, \ldots, q_{k-1})$ is induced by choosing a *single* value $\rho$ uniformly at random from $[0, 1/d]$, and then by taking a $[q_1 + \rho, q_2 + \rho, \ldots, q_{k-1} + \rho]$-sparc. We call this type of sub-distribution a *uniform sparc range.*

Let $\mathcal{Q}_u$ denote the set of uniform sparc ranges, and let $\mathcal{W}$ denote the same set of cells used by Karger et al. For our setup of LP-UB, we use $\mathcal{Q}_u$ as our sub-distributions and $\mathcal{W}$ as our cells. One can prove that, if $Q$ and $Q_u$ are a sparc range and uniform sparc range, respectively, defined by the same set of coordinates, then $\theta(Q, W) = \theta(Q_u, W)$. We use uniform sparc ranges, rather than sparc ranges, solely to simplify our computation of $\theta(Q, W)$. Again, we have that $\theta_{1,2}(Q, W) = \theta(Q, W)$, and so we focus our attention on the computation of $\theta_{1,2}(Q, W)$ for $Q \in \mathcal{Q}_u$ and

$$\begin{array}{rcl} \sigma & : & [7,3,\underline{1},4,\underline{2},6,5] \\ \sigma' & : & [7,3,\underline{2},4,\underline{1},6,5] \end{array}$$

Figure 3-1: A 1,2-conjugate pair of permutations, for the $k = 7$ case. Note that the two permutations are identical except for the positions of 1 and 2.

$W \in \mathcal{W}$.

As before, each uniform sparc range $Q$ can itself be thought of as a uniform probability distribution over $k!$ sub-distributions, one per permutation of $\{1, \ldots, k\}$. Alternatively, we can partition the set of permutations of $\{1, \ldots, k\}$ into $\frac{k!}{2}$ $1,2$-*conjugate pairs.* Two permutations $\sigma$ and $\sigma'$ are considered to form a $1,2$-conjugate pair if they differ only in that the positions of 1 and 2 are switched between them. Then, each uniform sparc range $Q$ can be thought of as a uniform probability distribution over $\frac{k!}{2}$ sub-distributions, one per $1,2$-conjugate pair.

Let $Q_{\sigma,\sigma'}$ be the sub-distribution corresponding to the permutation pair $(\sigma, \sigma')$. As before, we have that

$$\sup_{s \in S_{1,2}(W)} \sum_{(\sigma,\sigma')} \tau_k(Q_{\sigma,\sigma'}, s) \leq \sum_{(\sigma,\sigma')} \sup_{s \in S_{1,2}(W)} \tau_k(Q_{\sigma,\sigma'}, s) \tag{3.5}$$

Thus, we have

$$\begin{aligned} \theta_{1,2}(Q, W) &= \sup_{s \in S_{1,2}(W)} \frac{2}{k!} \sum_{(\sigma,\sigma')} \tau_k(Q_{\sigma,\sigma'}, s) \\ &\leq \sum_{(\sigma,\sigma')} \frac{2}{k!} \sup_{s \in S_{1,2}(W)} \tau_k(Q_{\sigma,\sigma'}, s) \\ &= \frac{2}{k!} \sum_{(\sigma,\sigma')} \theta_{1,2}(Q_{\sigma,\sigma'}, W). \end{aligned} \tag{3.6}$$

Here, the suprema are taken over $1,2$-aligned segments, and the sums are taken over all $1,2$-conjugate pairs.

Implicit in the work of Călinescu et al. is an exact computation of the maximum density $\tau_k(Q_{\sigma,\sigma'}) = \sup_{s\in\Delta} \tau_k(Q_{\sigma,\sigma'}, s)$ when $\sigma$ and $\sigma'$ differ in their relative ordering of 1 and 2. The value they compute for this quantity is 1.5. Since $\sigma$ and $\sigma'$ differ in their relative ordering of 1 and 2 when $(\sigma, \sigma')$ is a 1,2-conjugate pair, we are able to apply their analysis to compute the values of $\theta_{1,2}(Q_{\sigma,\sigma'}, W)$ for 1,2-conjugate pairs $(\sigma, \sigma')$.

Both $\sigma$ and $\sigma'$ are equally likely to be selected from $Q_{\sigma,\sigma'}$, so that for any fixed segment $s$,

$$\tau_k(Q_{\sigma,\sigma'}, s) = \frac{1}{2}\left(\tau_k(Q_\sigma, s) + \tau_k(Q_{\sigma'}, s)\right). \tag{3.7}$$

For $1,2$-conjugate pairs where only one of the slices for vertex 1 or vertex 2 can pass through the cell, the analysis of the previous section applies; we have that $\tau_k(Q_\sigma, s) = \tau_k(Q_{\sigma'}, s) = d$, and so

$$\theta_{1,2}(Q_{\sigma,\sigma'}, W) = \sup_{s\in S_{1,2}(W)} \tau_k(Q_{\sigma,\sigma'}, s) = d.$$

The interesting case is when $\sigma$ and $\sigma'$ are permutations such that both the slices for vertex 1 and vertex 2 can pass through the cell.

Consider a $1,2$-aligned segment $s$. Define the following quantities:

$$\begin{aligned} m_1 &= \min_{x\in s} x_1 - w_1 \\ M_1 &= \max_{x\in s} x_1 - w_1 \\ m_2 &= \min_{x\in s} x_2 - w_2 \\ M_2 &= \max_{x\in s} x_2 - w_2 \end{aligned}$$

Note that $|s| = M_1 - m_1 = M_2 - m_2$. Without loss of generality, we may assume that $M_1 \le m_2$. This is because a $1,2$-aligned segment can be split in two with one part closer to the hyperplane $x_1 = w_1$ and one part closer to the hyperplane $x_2 = w_2$, and our assumption then applies to each part separately (see Figure 3-2). Suppose that $\sigma$ is the permutation where 1 precedes 2, and that $\sigma'$ is the permutation where 2 precedes 1. Let us consider each of $\tau_k(Q_\sigma, s)$ and $\tau_k(Q_{\sigma'}, s)$ in turn.

First, it is easy to see that $\tau_k(Q_\sigma, s) = 2d$. This is because $s$ will be cut iff

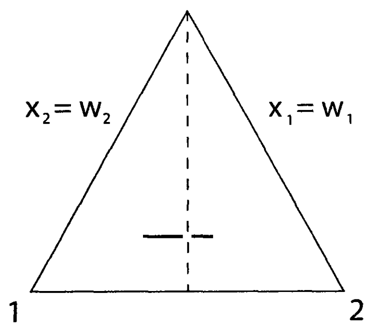

Figure 3-2: A 1,2-aligned segment, split into two parts, with one part closer to the hyperplane $x_1 = w_1$ and one part closer to the hyperplane $x_2 = w_2$. The dividing hyperplane, represented by the vertical dashed line, can be expressed as $x_1 - x_2 = w_1 - w_2$. In this case, the assumption that $M_1 \leq m_2$ holds for the part of the segment to the right of the dashed line.

$\rho \in [m_1, M_1] \cup [m_2, M_2]$, which is a subset of $[0, d]$ of measure $2|s|$. On the other hand, we can see that $\tau_k(Q_{\sigma'}, s) = d$, since $s$ will be cut iff $\rho \in [m_2, M_2]$, a subset of $[0, d]$ of measure $|s|$. The reason that $s$ will not be cut if $\rho \in [m_1, M_1]$ is that the slice for vertex 2 will capture all of $s$ before the slice for vertex 1 has a chance to cut it. Figure 3-3 summarizes this argument.

Plugging into (3.7), then, we have that $\tau_k(Q_{\sigma,\sigma'}, s) = 1.5d$, and thus that

$$\theta_{1,2}(Q_{\sigma,\sigma'}, W) = \sup_{s \in S_{1,2}(W)} \tau_k(Q_{\sigma,\sigma'}, s) = 1.5d.$$

If we define $\chi$ as in the previous section, this analysis shows:

$$\theta_{1,2}(Q_{\sigma,\sigma'}, W) = \begin{cases} 1.5d & \text{if } \chi(Q_\sigma, W) = 2 \\ \frac{d}{2} \cdot (\chi(Q_\sigma, W) + \chi(Q_{\sigma'}, W)) & \text{else.} \end{cases}$$

Combining with (3.6) gives

$$\theta(Q, W) \leq \frac{1}{k!} \sum_\sigma g(\chi(Q_\sigma, W)),$$

where $g(x) = 1.5$ if $x = 2$ and $g(x) = x$ otherwise.

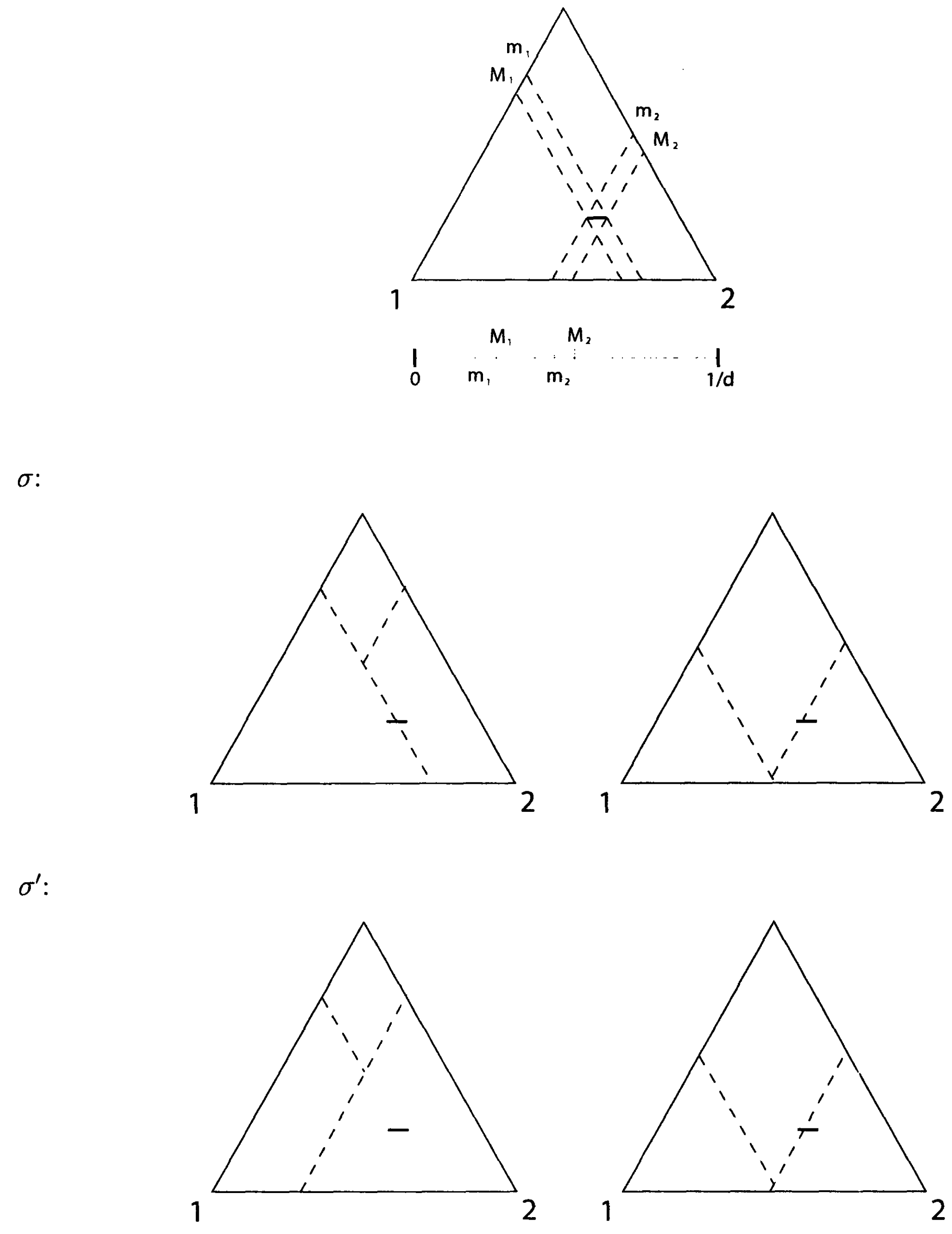

Figure 3-3: A summary of the density contributions from a (1,2)-conjugate pair sub-distribution. At the top, we have a cell with a 1,2-aligned segment obeying the constraint $M_1 \leq m_2$. Directly below is the $1/d$-length interval $[0, 1/d]$, with the disjoint sub-intervals $[m_1, M_1]$ and $[m_2, M_2]$ highlighted. The next row shows partitions generated by $\sigma$, and the bottom row shows partitions generated by $\sigma'$. In both rows, the left side shows the case when $\rho \in [m_1, M_1]$, and the right side shows the case when $\rho \in [m_2, M_2]$. The key observation is that the segment is not cut when the permutation is $\sigma'$ and when $\rho \in [m_1, M_2]$.

In fact, our analysis demonstrates that for segments $s \in S_{1,2}(W)$, with the property that $M_1 \leq m_2$, $\tau_k(Q_{\sigma,\sigma'}, s)$ exactly equals $\theta_{1,2}(Q_{\sigma,\sigma'}, W)$ for all $Q \in \mathcal{Q}_u$ and all 1, 2-conjugate pairs $(\sigma, \sigma')$. Thus, the inequality of (3.5) is tight, implying that

$$\theta(Q, W) = \sum_{\sigma} g(\chi(Q_\sigma, W)). \tag{3.8}$$

Note that without $g$, the right hand side simply becomes the density bound of Karger et al. given in (3.4).

Moreover, this implies that the inequality in (2.1), back in Section 2.2.3, is tight for the same reason. Thus, by substituting these values of $\theta(Q, W)$ for $\psi(Q, W)$ in LP-UB, we obtain a linear program equivalent to the linear program PRI-$\infty_2$, allowing us to apply Theorem 5. The implication is that our analysis of the performance of the cutting schemes outputted by our computational experiments is exact.

## 3.3 Observations and results

We wrote a simple program to generate the linear programs described in this chapter automatically, and used CPLEX to solve them. While it is difficult to "prove" programs correct, our computations reproduced the results of Karger et al. prior to incorporating the function $g$ in Equation (3.8), giving us confidence in the program's correctness.

Table 3.1 summarizes the results of our computational experiments. Under the assumption that the programs were correct, the listed bounds are proven upper bounds on the integrality gap. The cutting schemes outputted by the program have maximum density exactly equal to the given bounds. One unusual phenomenon we noticed was the corner cut probability did not decrease monotonically with respect to $k$ as it had when using the Karger et al. bounds. In particular, the corner cut probability for $k = 7$ was higher than for other values of $k$. We have no hypothesis for why this might be the case. Perhaps the notion of corner cuts is not as significant as was originally suspected. In other respects, we did not notice any qualitative difference

| $k$ | discretization level | Karger et al. bounds | | Improved bounds | |
|---|---|---|---|---|---|
| | | corner cut prob. | bound | corner cut prob. | bound |
| 4 | 36 | .289 | 1.1539 | .299 | 1.1494 |
| 5 | 18 | .314 | 1.2161 | .326 | 1.2071 |
| 6 | 12 | .376 | 1.2714 | .360 | 1.2573 |
| 7 | 9 | .397 | 1.3200 | .284 | 1.2967 |

Table 3.1: The upper bounds on the integrality gap obtained by improving the analysis of Karger et al.

in the solution to the linear program as a result of modifying the bounds.

# Chapter 4

# Lower Bounding the Integrality Gap

In this chapter, we present some approaches used to lower bound the value of $\tau_k^*$. Our approach has successfully generated computer-aided proofs of lower bounds on $\tau_k^*$, as shown in Table 4.1 near the end of the chapter.

## 4.1 Preliminaries

Recall that $w_0(G)$ denotes the cost of the minimum $k$-way cut of $G$. For any embedding, $\alpha$, of $G$ into $\Delta_k$, let $\gamma(G, \alpha) = \frac{w_0(G)}{\text{vol}(\alpha(G))}$. Let $\alpha^*(G)$ denote the minimum volume embedding of $G$ into $\Delta_k$, and let $\gamma(G) = \gamma(G, \alpha^*(G))$ for all $G$. Note that $\gamma(G, \alpha) \leq \gamma(G)$ for all embeddings $\alpha$.

Theorem 2 tells us that $\tau_k^*$ is a value such that $\gamma(G) \leq \tau_k^*$ for all $G$. In order to prove a lower bound of $\tau_0$ on $\tau_k^*$, then, it suffices to demonstrate a particular graph $G_0$ and a particular embedding $\alpha_0$ such that $\gamma(G_0, \alpha_0) \geq \tau_0$. In fact, Theorem 2 guarantees the existence of such an embedded graph for any fixed $\tau_0 < \tau_k^*$, motivating us to search for bad embedded graphs.

## 4.2 Finding exact minimum $k$-way cuts

We wish to find embedded graphs $\alpha(G)$ for which the quantity $\gamma(G, \alpha) = \frac{w_0(G)}{\mathrm{vol}(\alpha(G))}$ is large. In order to compute this value, we need to compute the quantity $w_0(G)$. Unfortunately, finding the minimum $k$-way cut is a MAX-SNP hard problem (which is the reason we are performing this study in the first place). Thus, we set out to devise exact algorithms to solve minimum multiway cut efficiently in practice. Our approach uses the typical branch-and-bound paradigm.

In the remainder of this section, we will revert to using *vertices* to describe the elements of $V$. *Nodes* will refer to the elements of the enumeration tree.

### 4.2.1 Branch and bound enumeration tree

Note that any minimal $k$-way cut of a connected graph leaves every vertex connected to a unique terminal; i.e., it never disconnects a vertex from all the terminals. Thus, any minimal $k$-way cut of a connected graph can be thought of as an assignment of vertices to terminals, $f : V \to T$, such that $f(t) = t \; \forall t \in T$. Conversely, any such assignment corresponds to a $k$-way cut of the graph: take as the cut the set of all edges $(u, v)$ for which $f(u) \neq f(v)$. In this context, the cost of a cut is the sum of the weights of the edges $(u, v)$ such that $f(u) \neq f(v)$.

This leads to the following equivalent phrasing of the minimum multiway cut problem: given a weighted graph $G = (V, E)$, together with a set $T \subseteq V$, find an assignment $f : V \to T$, with $f(t) = t$ for all $t \in T$, that minimizes the quantity $\sum_{e \in L} w(e)$, where $L = \{(u, v) \in E : f(u) \neq f(v)\}$. We let $t_1, t_2, \ldots, t_k$ be the elements of $T$. This is in fact an instance of the more general *discrete metric labeling problem*, abbreviated DMLP. A DMLP branch-and-bound heuristic is described in [10], and our approach is based on it.

In the branch-and-bound enumeration tree, node $i$ of the enumeration tree represents a partial assignment, $f_i : S_i \to T$, of the vertices of some subset $S_i \subseteq V$ such that $T \subseteq S_i$, and such that $f_i(t) = t \; \forall t \in T$. The root node, $r$, is associated with the partial assignment where $S_r = T$. The children of any node $i$ of the enumeration tree are obtained as follows. Pick a vertex $u \in V - S_i$ and generate $k$ children of $i$. Let $j(1), j(2), \ldots, j(k)$ be the children of $i$ in the enumeration tree. For each $j(l)$, $f_{j(l)}$ is an extension of $f_i$. More precisely, $f_{j(l)} : S_i \cup \{u\} \rightarrow T$, with $f_{j(l)}(u) = t_l$. Then, any child of $i$ represents a partial assignment of $S_i \cup \{u\}$.

Note that the leaves of this tree represent complete assignments. For each leaf $\ell$ let $val(\ell)$ be the cost of the $k$-way cut represented by $\ell$. We wish to find the leaf $\ell$ for which $val(\ell)$ is minimized.

There are a couple details to address to complete the description of this branch-and-bound algorithm:

- **Lower bounds**: given an internal tree node, $i$, find a good bound $b(i)$ such that $val(\ell) \geq b(i)$ for all leaf-descendents $\ell$ of $i$.
- **Branching order**: determine which node to process next.

### 4.2.2 Lower bounds

In order to make our branch-and-bound algorithm run efficiently, we need, for any node $i$, a good bound $b(i)$ such that $val(\ell) \geq b(i)$ for all leaf-descendents $\ell$ of $i$. The reason this is desirable is that if $b(i)$ is greater than the cost of an already-discovered $k$-way cut, it allows us to skip over the entire sub-tree (a technique known as *pruning*). In our context, we seek a lower bound on the value of any $k$-way cut generated by an extension of a given partial assignment $f_i : S_i \rightarrow T$.

To this end, consider the graph, $G_i$, formed from $S_i$ and $f_i$ by contracting each $u \in S_i$ into the terminal $f_i(u)$. Note that any $k$-way cut of $G$ induced by the partial coloring $f_i$ corresponds to a $k$-way cut of $G_i$. It suffices to find a lower bound on the value of any $k$-way cut of $G_i$.

For this, we can use any deterministic constant-factor approximation algorithm that approximately solves minimum multiway-cut on $G_i$. If an $\alpha$-approximation algorithm is used, and the resulting $k$-way cut found has value $x$, then the lower bound would be $x/\alpha$. In fact, we can do much better if there are heavy weight edges between pairs of terminals in $G_i$, since those edges must be cut by any $k$-way cut of $G_i$.

Because of this, we first remove all edges between pairs of terminals in $G_i$ and use the approximation algorithm on the resulting graph to obtain a cut of value $y$. Then, if we let $z$ be the total weight of these removed edges, our lower bound is $z + y/\alpha$. Without this heuristic, the lower bound would simply be $(z + y)/\alpha$, a smaller quantity. A second refinement is based on the observation that if some vertex is adjacent to each of the $k$ contracted terminals, then exactly $(k-1)$ of those connecting edges must be cut. If we let $w$ be the weight of the minimum weight edge connecting the vertex to the contracted terminals, we can increment $z$ by $(k-1)w$ and decrement each of those edge weights by $w$. A similar argument shows that this heuristic results in a better lower bound.

As for how to obtain a constant-factor approximation to the minimum $k$-way cut of $G_i$, there are a few options. One option is to use the $(2-2/k)$-approximation algorithm of Dahlhaus et al. (described in Section 1.2). Another is to use the Călinescu et al. embedding and to take the minimum sparc-induced $k$-way cut, which achieves the bounds shown in Table 3.1. However, the bound produced by the Dahlhaus et al. scheme is relatively weak, while finding the minimum sparc-induced cut after applying the Călinescu et al. embedding is too costly, as it requires the enumeration of $O(k! \cdot n^{k-1})$ sparcs. The middle ground we chose was a derandomization of the randomized algorithm given by Călinescu et al. Their algorithm involves choosing a single value $\rho$ uniformly at randomly from $[0, 1]$ and choosing either the identity permutation or the permutation $\sigma$ such that $\sigma(k) = k$ and $\sigma(i) = k - i$ for all $i < k$, each with probability $1/2$. It then performs the sparc given by setting all the slicing distances equal to $\rho$ and by ordering the terminals according to the selected permutation. This achieves an expected approximation guarantee of $3/2 - 1/k$. We enumerated all $O(n)$ possible cuts generated in this manner, together with their respective cut values and probabilities, to compute the expected cut value exactly. Note that this value is preferable to the minimum of the $O(n)$ possible cuts, as it produces a larger lower bound.

### 4.2.3 Branching order

In order to specify a branching order, we first fix an ordering on the non-terminal vertices, so that each level of the tree corresponds to a non-terminal vertex. Preferably, this ordering should be such that there is a good chance that some subtrees rooted high up in the tree can be pruned during the course of the computation.

We captured this idea by computing, for each non-terminal vertex $v$, the weight of the minimum path from $v$ to each of the terminals. We let the *rank* of $v$ equal the difference between the maximum and the minimum of the $k$ resulting values. The vertex ordering was performed in decreasing order of rank. The intuition behind this heuristic is that a vertex that is close to one terminal but far from some others should probably be assigned to the closer terminal, so we expect the other branches to be cut off relatively quickly.

This resolves the question of how to order the non-terminal vertices of the graph. In other words, it matches each level of the branch-and-bound tree with a vertex of the graph. We must still answer the question of how to choose which (tree) node to process among a set of active nodes. The heuristic we chose to use was to compute, for each node $i$, the lower bounds $b(j)$ for each child $j$ of $i$. The next node to process is then the node $i$ for which the quantity $\max\{b(j) - b(j') \mid j, j' \text{ children of } i\}$ is largest. The intuition behind this heuristic is similar to that of the rank notion used for vertex ordering. If two children of a node differ widely in their lower bounds, the hope is that the child with the small lower bound will lead to the discovery of a small cut, which allows the subtree rooted at the child with the large lower bound to be pruned.

### 4.2.4 Empirical performance

For all our clever scheming, we found that our Java implementation of this algorithm only offered a slight improvement in performance over the straightforward approach of feeding CPLEX an integer program. One way to formulate the minimum multiway cut problem as an integer program, for instance, is to define a 0,1-valued distance variable between pairs of vertices, and to impose that the distance variables obey a metric and that terminals lie at distance 1 from each other. The quantity to be minimized is then the sum $\sum_{u,v} d(u,v)w(u,v)$, where $d$ and $w$ represent the distance and weight functions, respectively.

This came as a surprise to us, and perhaps is a testament to the power of the CPLEX software. Still, we believe our branch-and-bound algorithm to have some potential value - that with some better heuristics and a better implementation, it should be capable of outperforming a generic integer program solver.

## 4.3 Searching for bad embedded graphs

Section 4.2 shows how one can compute the minimum $k$-way cut of a small graph exactly. With this procedure, then, we can evaluate the quantity $\gamma(G, \alpha)$ for small embedded graphs $\alpha(G)$. In this section, we describe one approach we used to search for small embedded graphs, $\alpha(G)$, for which $\gamma(G, \alpha)$ was large. This approach led to the lower bounds shown in Table 4.1, which represent the best known lower bounds to date.

### 4.3.1 Grid graphs

The set of all embedded graphs represents a large search space. We restricted our search by fixing an integer *discretization level d*, and by considering the *size-d grid graph.* The size-$d$ grid graph is an embedded graph that can be described as follows: the embedded nodes consist of all points in the simplex whose coordinates are multiples of $1/d$, and the embedded edges consist of all segments connecting embedded nodes that are at distance $1/d$ from each other. A combinatorial argument shows that this embedded graph has $n = \binom{d+k-1}{k-1}$ nodes and $m = \binom{k}{2} \cdot \binom{d+k-2}{k-1} = \theta(k^2 n)$ edges. Our search space was then the set of weight functions $w : E_d \rightarrow \mathbb{R}$, where $E_d$ is the set of edges of the size-$d$ grid graph.

With this particular embedding $\alpha_d$, our task is to maximize the quantity

$$\gamma(G, \alpha_d) = \frac{w_0(G)}{\mathrm{vol}(\alpha_d(G))}.$$

Since scaling $w$ by a constant does not change the value of $\gamma$, we restricted our search to those $w$ for which $\text{vol}(\alpha_d(G)) = 1$. We could then maximize $\gamma(G, \alpha_d)$ and thus find the optimal $w$ by solving the following linear program:

$$\begin{aligned} &\max \lambda \ s.t. \\ &\sum_s w_s \cdot \frac{1}{d} = 1 \\ &\sum_{s \in C} w_s \geq \lambda \qquad \forall C \in \mathcal{D}^* \end{aligned}$$

Here, $\mathcal{D}^*$ denotes the set of all $k$-way cuts of $G$.

We can also take advantage of the symmetry of $\alpha_d$ to impose further constraints. Specifically, the symmetry of $\alpha_d$ allows us to assume that our weight function is also symmetric; i.e., that the edge weights remain invariant under any of the $k!$ symmetric transformations of the simplex. To see this, consider an arbitrary weight function $w$ such that $\sum_s w_s \cdot \frac{1}{d} = 1$. Now let $w^*$ be the weight function that assigns to edge $e$ the average of $w'(e)$ taken over all $k!$ symmetric versions of $w$. It is clear that $w^*$ also satisfies $\sum_s w_s \cdot \frac{1}{d} = 1$, and that it is symmetric. To complete the argument, consider any cut $C$. Then, $w^*(C)$ is equal to the average of $w'(C)$ where $w'$ is taken over all symmetric versions of $w$. Alternately, this can be expressed as the average of $w(C')$ where $C'$ is taken over all symmetric versions of $C$. This quantity can be bounded below by the value of the minimum cut with respect to $w$. It follows that $w^*(C)$ is at least as large as the cost of the minimum cut with respect to $w$, from which our claim follows. With our symmetry argument, our new linear program is:

$$\begin{aligned} &\max \lambda \ s.t. \\ &\sum_s w_s \cdot \frac{1}{d} = 1 \\ &\sum_{s \in C} w_s \geq \lambda \qquad \forall C \in \mathcal{D}^* \\ &w_s = w_{s'} \qquad \forall s \sim s' \end{aligned} \tag{LP2}$$

Here, $s \sim s'$ means that $s$ and $s'$ are equivalent under one of the $k!$ symmetries of the simplex. By adding these symmetry constraints, we effectively reduce the number of weight variables by a factor of $O(k!)$.

Unfortunately, even with this optimization, the linear program has exponentially many constraints. As thus, we could not hope to fit LP2 into memory for reasonably large $n$. On the other hand, we only expected a small fraction of the constraints to be binding[1] in an optimal solution, so we turned our efforts towards finding a small subset of $\mathcal{D}^*$ which corresponded to binding constraints. To this end, we considered the following linear program, parameterized by a subset $\mathcal{D} \subseteq \mathcal{D}^*$:

$$\begin{aligned} &\max \lambda \ s.t. \\ &\sum_s w_s \cdot \frac{1}{d} = 1 \\ &\sum_{s \in C} w_s \geq \lambda \qquad \forall C \in \mathcal{D} \\ &w_s = w_{s'} \qquad \forall s \sim s' \end{aligned} \tag{LP2($\mathcal{D}$)}$$

Let $(\lambda^{(\mathcal{D})}, w^{(\mathcal{D})})$ be an optimal solution of the linear program LP2($\mathcal{D}$). We reiterate our goal more precisely: to find a small subset $\mathcal{D} \subseteq \mathcal{D}^*$ such that $\lambda^{(\mathcal{D})} = \lambda^{(\mathcal{D}^*)}$.

We initialized $\mathcal{D}$ to some proper subset of $\mathcal{D}^*$, and solved LP2($\mathcal{D}$) to find $(\lambda^{(\mathcal{D})}, w^{(\mathcal{D})})$. We then updated our graph weights by setting $w \leftarrow w^{(\mathcal{D})}$. Next, we found a violating constraint of LP2 - i.e., a $k$-way cut $C \in \mathcal{D}^* - \mathcal{D}$ such that $w^{(\mathcal{D})}(C) < \lambda^{(\mathcal{D})}$. We set $\mathcal{D} \leftarrow \mathcal{D} \cup \{C\}$, and repeated. When no more violating constraints could be found, the final value of $\lambda^{(\mathcal{D})}$ would be equal to our desired value of $\lambda^{(\mathcal{D}^*)}$.

The method just described is a well-known technique in linear programming. It is called *column generation*, and is commonly used to identify the binding constraints of large linear programs, typically in hopes of reducing the problem instance to a more computationally manageable size. This is precisely the problem we have at hand. Column generation was pioneered by Gilmore and Gomory in 1961 [7]; a survey of column generation techniques and applications can be found in [9].

There are still a couple issues to resolve in our particular context. First, how are

[1]In linear programming, a constraint is *binding* if the removal of the constraint changes the optimum value of the program

we to find violating constraints? Second, how should $\mathcal{D}$ be initialized? The second issue is of interest, because if finding violating constraints is expensive, we would want a large initial $\mathcal{D}$ in order to minimize the number of times a violating constraint had to be found.

We begin by addressing the second issue. Recall that the embedding $\alpha$ is fixed in this setting. We chose to initialize $\mathcal{D}$ to the set of all $O(k! \cdot n^{k-1})$ sparc-induced cuts. This seemed to work well in practice.

As for the first issue, we are again faced with the problem of solving an NP hard problem, as finding the minimum $k$-way cut reduces to the NP hard decision problem of determining whether there exists a $k$-way cut of weight $\leq \lambda$. Again, we used our branch-and-bound strategy to solve this problem in a reasonable amount of time. Note that we have a choice: we can stop as soon as a cut of weight $\leq \lambda$ is found, or we can search for the minimum cut. We chose the first option, as it was much faster.

The following summarizes our procedure for finding the weight function $w$ that minimizes $\gamma$:

1. Initialize $\mathcal{D}$ to the set of all sparc-induced $k$-way cuts of $G$.

2. Solve the linear program LP2($\mathcal{D}$) to find $w^{(\mathcal{D})}$ and $\lambda^{(\mathcal{D})}$.

3. Use the branch-and-bound strategy outlined in Section 4.2 to find a violating cut $C$, if one exists. If a cut is found, set $\mathcal{D} \leftarrow \mathcal{D} \cup \{C\}$, and return to step 2. Otherwise, proceed to step 4.

4. Return $w^{(\mathcal{D})}$ as the optimal weight function $w$.

### 4.3.2 General embedded graphs

Grid graphs are computationally convenient: they exhibit symmetry and only have $O(n)$ edges. The symmetry property is especially important in our column generation paradigm, since the discovery of a single violating cut effectively leads to the discovery of $O(k!)$ violating cuts, one for each symmetric version. This greatly reduces the number of column generation iterations.

Nevertheless, one must ask the question of whether an arbitrary embedded graph can achieve a greater value of $\gamma(G, \alpha)$. We thus sacrificed some of the computational simplifications of grid graphs and generalized our experiment to arbitrary embeddings. For a given embedding $\alpha$, this generalization was straight-forward: we removed the edge weight symmetry constraint from the various linear programs, and replaced the constraint

$$\sum_s w_s \cdot \frac{1}{d} = 1$$

everywhere with the more general constraint

$$\sum_s w_s \cdot |s| = 1.$$

We also worked with complete graphs for maximum generality, since a missing edge can be modeled by an edge with weight 0. Our search space was then the set of weight functions $w : E \to \mathbb{R}$, together with the set of embeddings $\alpha$.

We then chose initial values for $w$ and $\alpha$ and did the following:

1. Fix $w$, and solve for the embedding $\alpha$ that maximizes $\gamma(G, \alpha)$ (i.e., minimize the volume of the embedding).

2. Fix $\alpha$, and solve for the weight function $w$ that maximizes $\gamma(G, \alpha)$ (i.e., induce the worst possible ratio).

3. Repeat 1-2 until $w$ and $\alpha$ remain fixed.

Performing step 1 is straightforward; one simply needs to solve the linear program LP. Performing step 2 was the subject of the prior section.

### 4.3.3 Results and Observations

The algorithm described in this chapter was implemented via a combination of Java and CPLEX. While it is difficult to prove every part of the program correct, it is possible to independently compute both the value of the minimum multiway cut of the output graph and the value of the embedding, and to compare the quotient of these

| $k$ | $n$ | bound | previous bound |
|---|---|---|---|
| 4 | 35 | 1.1052 | $12/11 \approx 1.0909$ |
| 5 | 35 | 1.1098 | $32/29 \approx 1.1034$ |

Table 4.1: Improved lower bounds on the integrality gap.

values with the value of $\gamma(G, \alpha)$ outputted by the program. We could have confidence in the correctness of the minimum multiway cut implementation because we had two different implementations (the branch-and-bound algorithm and the integer program) that agreed on the given inputs. We could have confidence in the correctness of the embedding implementation because of the simplicity of the program, and by hand-checking the program outputs for small inputs by hand.

The size of the graphs we could consider was severely restricted by the fact that our procedure's running time is theoretically asymptotically exponential in $n$. Nevertheless, we were able to run our algorithm on large enough instances to obtain new lower bounds on the integrality gap for small values of $k$. The previously best known lower bound for $k \geq 4$ was $8/(7 + \frac{1}{k-1})$, discovered by Freund and Karloff [6]. Our results are summarized in Table 4.1. Unfortunately, for $k > 5$, we could not run our algorithm on large enough instances to generate improved lower bounds.

There are some interesting observations to point out. We found that the generalized experiment of Section 4.3.2 offered no improvement over the grid graph-based experiment of Section 4.3.1. When our input graph had the same number of nodes as the size-$d$ grid graph for some $d$, the generalized algorithm typically converged, in just a few iterations, to the size-$d$ grid graph, regardless of the initial weight function or embedding. The only other local optima we could discover in the search space were somewhat pathological examples (such as embeddings where two embedded nodes coincide). When our input graph was not of the appropriate size, the final embedding was such that each embedded node sat at the location of an embedded node of the next largest size-$d$ grid graph. This seems to provide empirical evidence indicating that the grid graph is optimal among all possible worst-case embeddings.

Finally, we should comment on the nature of the cuts corresponding to the violating constraints we found. For the $k = 5$ case, we found no violating constraints. For the $k = 4$ case, violating constraints were found; we wrote a program to help visualize these cuts, in hopes of finding some obvious structure that would allow us to extend these cuts into a class of $k$-way cuts of the simplex. A typical cut is shown in Figure 4-1. Unfortunately, we could find no such structure - these cuts could not be expressible as sparcs, pair-isolating cuts, or any other sort of “natural” cut that we could conceive of. Perhaps if we could solve much larger instances of the problem, patterns may have emerged. The MAX-SNP hardness of the computation of $\gamma(G)$, however, may make this an impractical line of future research. Furthermore, one cannot logically infer that these violating cuts correspond to cuts that are used by the optimal cutting scheme, possibly rendering such an investigation moot. They could simply be an artifact of the discretization. More will be said of this latter point in Chapter 5.

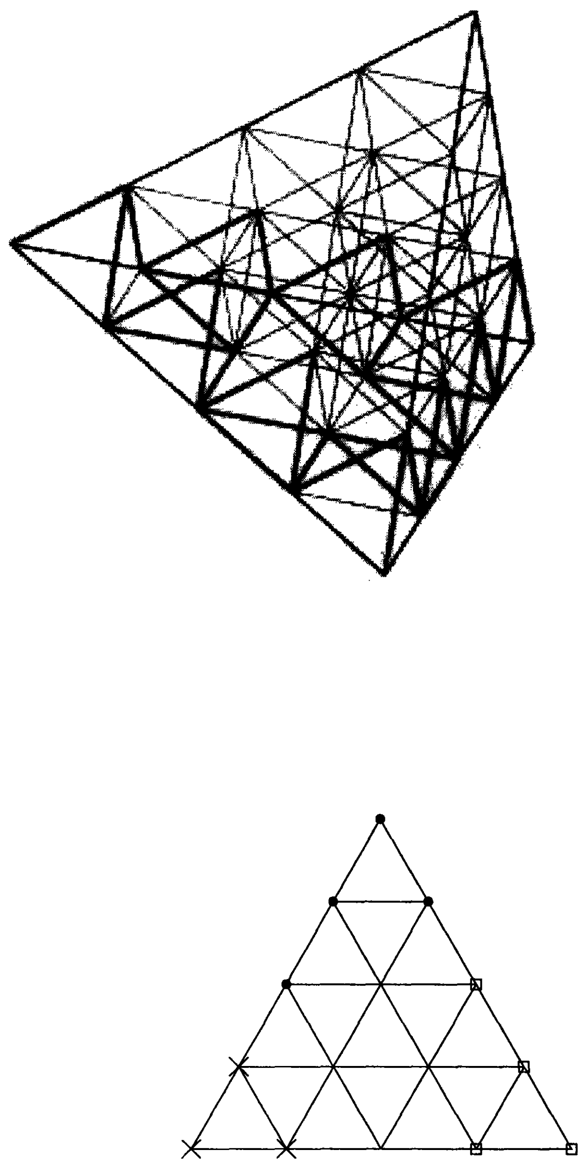

Figure 4-1: A graphical representation of a non-sparc cut corresponding to a binding constraint for the $k = 4$ case. The edges belonging to the cut are highlighted in the three dimensional representation on top. Note that the edges belonging to the cut either lie completely within the bottom face or are incident to the bottom face. The bottom face is shown on bottom, with the nodes labeled by the assigned terminal. The empty label corresponds to the opposite terminal; all other nodes of the simplex would share this label if they were depicted.

# Chapter 5

# Study of Pair-Isolating Cuts

Aoyama showed that for a fixed discretization level, the optimal cutting distribution using sparcs and pair-isolating cuts yielded a smaller maximum density than the optimal cutting distribution using simply sparcs. As a result, he postulated that an optimal cutting scheme for $k = 4$ may not be describable as a probability distribution over sparcs, although he recognized that this postulate could not be logically inferred from his computational experiments.

We have performed some additional computational experiments on the $k = 4$ case to further investigate this postulate. Although our results cannot confirm or deny his postulate, they seem to suggest that the addition of pair-isolating cuts in fact does *not* yield a better cutting scheme.

## 5.1 Interpreting Mosk-Aoyama's results

Mosk-Aoyama's computational experiments seem to indicate that a cutting scheme involving sparcs and pair-isolating cuts can achieve a smaller maximum density than a cutting scheme involving just sparcs. Although he mentions that the postulate did not necessarily follow from his computational results, we offer a more in-depth analysis of why this need not be the case, based on our discussion from section 2.2.3.

Recall the formulation of the linear program LP-UB. We have sub-distributions $\{Q_i\}$, cells $\{W_j\}$, and density bounds $\{\psi(Q_i, W_j)\}$, organized as follows:

$$\min \tau$$
$$\sum_i p_i = 1$$
$$\sum_i p_i \cdot \psi(Q_i, W_j) \leq \tau \qquad \forall W_j$$

Let $\mathcal{Q}$, $\mathcal{W}$, and $\Psi$ represent the sets $\{Q_i\}$, $\{W_j\}$, and $\{\psi(Q_i, W_j)\}$, respectively. We index each of these sets by the discretization level ($d$) used and whether they are the sets used in the framework of Karger et al. ($K$) or Mosk-Aoyama ($M$):

$$\begin{array}{ccc} \mathcal{Q}_d^{(K)} & \mathcal{W}_d^{(K)} & \Psi_d^{(K)} \\ \mathcal{Q}_d^{(M)} & \mathcal{W}_d^{(M)} & \Psi_d^{(M)} \end{array}$$

Additionally, let $\tau_d^{(K)}$ and $\tau_d^{(M)}$ represent the objective values of LP-UB under the Karger et al. and Mosk-Aoyama framework, respectively.

Mosk-Aoyama's postulate was borne from the observation that, for small fixed $d$,

$$\tau_d^{(K)} > \tau_d^{(M)}. \tag{5.1}$$

A problem with this reasoning, however, is that $\mathcal{W}_d^{(M)}$ represents a (strictly) finer discretization of the simplex than $\mathcal{W}_d^{(K)}$ does - each cell of $\mathcal{W}_d^{(K)}$ can be expressed as a union of multiple cells of $\mathcal{W}_d^{(M)}$. This fact may be slightly obscured by the parametrization of the sets of cells by the same value of $d$. It is thus not clear how to compare the amount of slack lost in inequality (2.1) by the density bounds $\Psi_d^{(K)}$ and $\Psi_d^{(M)}$, respectively.

As a result, the empirical observation of (5.1) could arise from the sub-distributions of $\mathcal{Q}_d^{(M)} - \mathcal{Q}_d^{(K)}$, from the granularity difference between $\mathcal{W}_d^{(M)}$ and $\mathcal{W}_d^{(K)}$, or from the looseness of the bounds $\Psi_d^{(K)}$ (or some combination of the three). Even if the cause of the empirical observation could somehow be wholly pinned to the first, it may be the case, as Mosk-Aoyama observes, that the difference between $\tau_d^{(K)}$ and $\tau_d^{(M)}$ approaches 0 as $d \to \infty$. The presence of these conflicting factors makes it difficult to have confidence in Mosk-Aoyama's postulate.

## 5.2 An alternative approach

The discussion of the previous section reveals an inherent difficulty with an approach that relies on the linear program LP-UB to try to reason about the roles of certain types of cuts in an optimal cutting scheme. The difficulty lies in the task of comparing the amount of slack lost in inequality (2.1) when using different sets of sub-distributions and cells.

For this reason, we performed computational experiments based on the linear program LP-LB in an effort to gain additional insight concerning the optimal cutting scheme without having to deal with the complication of slack-losing inequalities like (2.1).

Recall the formulation of LP-LB. We have a finite set $V \subseteq \Delta$, and a graph, $G(V)$, whose vertices are the points of $V$, and whose edges are the elements of $S(V)$. We let cuts$(V)$ denote the set of $k$-way cuts of $G(V)$. The linear program is then:

$$\begin{aligned} &\max \lambda \\ &\sum_{s \in S(V)} w_s \cdot |s| = 1 \\ &\sum_{s \in C} w_s \geq \lambda \qquad \forall C \in \text{cuts}(V) \end{aligned} \tag{LP-LB}$$

A natural choice for our embedded graph is the size-$d$ grid graph. As noted in Section 4.3.1, this graph allows us to add symmetry constraints that effectively reduce the number of weight variables. Furthermore, as noted in Section 4.3.3, we have empirical evidence that seems to indicate that the grid graph is optimal among all possible worst-case embeddings.

If some constraints induced by pair-isolating cuts turn out to be binding constraints, this would seem to indicate, by complementary slackeness, that an optimal cutting scheme should assign nonzero probability to pair-isolating cuts. Conversely, if none of those constraints are binding constraints, this would seem to indicate that some optimal cutting scheme may assign zero probability to pair-isolating cuts. Unfortunately, this intuition cannot be made rigorous. The reason is that complentary slackeness can only be applied to the (undiscretized) infinite linear programs PRI-$\infty$ and DUAL-$\infty$, as these linear programs are truly duals of each other. The linear program at hand, LP-LB, is only a discretization of DUAL-$\infty$; its dual has no obvious relation to PRI-$\infty$. The relationships between these linear programs were given in Figure 2-1, which we repeat below:

$$\begin{tikzcd}
\boxed{\text{PRI-}\infty} \arrow[r, leftrightarrow, "\text{dual}"] \arrow[d, "\text{discretize}"'] & \boxed{\text{DUAL-}\infty} \arrow[dd, "\text{discretize}"] \\
\boxed{\text{PRI-}\infty_2} \arrow[d, "\text{slack}"'] & \\
\boxed{\text{LP-UB}} & \boxed{\text{LP-LB}}
\end{tikzcd}$$

We allowed this pseudo-argument to guide our experiments, despite the fact that the intuition behind it could not be made rigorous.

As mentioned at the end of Section 4.3.3, when using the size-$d$ discretization grid for small values of $d$, we found that none of the cuts corresponding to the binding constraints of LP-LB could be expressed as pair-isolating cuts. Due to computational limitations, however, we could only verify this for $d \leq 4$. This, in itself, did not make for very compelling evidence against the utility of pair-isolating cuts.

To get around the problem of computational infeasibility, we replaced the set cuts$(V)$ in LP-LB with a smaller set of cuts, and investigated whether any of the binding constraints of the resultant linear program corresponded to pair-isolating cuts. This modified linear program is given in Section 4.3.1; we repeat it here:

$$\begin{aligned}
&\max \lambda \ s.t. \\
&\sum_{s} w_s \cdot \frac{1}{d} = 1 \\
&\sum_{s \in C} w_s \geq \lambda \qquad \forall C \in \mathcal{D} \\
&w_s = w_{s'} \qquad \forall s \sim s'
\end{aligned} \tag{LP2($\mathcal{D}$)}$$

Specifically, we felt that it would be useful to compare the objective values of LP2($\mathcal{D}$) obtained by setting $\mathcal{D}$ equal to

1. $\mathcal{S}$, the set of all sparc-induced cuts

2. $\mathcal{S} \cup \mathcal{P}$, the set of all sparc-induced cuts, together with the set of all pair-isolating cut-induced cuts

If the second objective value is equal to the first, this would imply that the constraints corresponding to pair-isolating cuts in the second linear program are not binding. This would seem to imply that those same cuts are not binding in LP-LB. However, we must note a hole in our pseudo-argument here: it is not generally true that a non-binding constraint of a linear program remains non-binding when additional constraints are added. See, for instance, Figure 5-1.

It is important to realize that the resultant objective values are *not* provable lower bounds on the integrality gap $\tau_k$. Provable lower bounds can be obtained by proceeding with column generation to generate a linear program equivalent to LP-LB. This was done in Chapter 4, but the experiment presented here is of a different nature.

Nevertheless, we proceeded with this doubly-holed experiment.

## 5.3 Experimental variations

Recall our definition of pair-isolating cuts: cuts that first isolate two pairs of vertices from each other, then separate each pair of vertices. This represents a broad class of cuts. Two particular sub-classes of interest are the set of pair-side cuts and pair-edge cuts (defined in Section 2.3). Our most basic implementation of the experiment described in the previous section took the set of cuts to be those induced by sparcs, by pair-side cuts, and by pair-edge cuts.

We then created a variant of this experiment that considers a much broader class of pair-isolating cuts. Namely, it considers those cuts generated by performing a pair-isolating slice (as defined in Section 2.3), followed by *any* arbitrary cut of the resulting

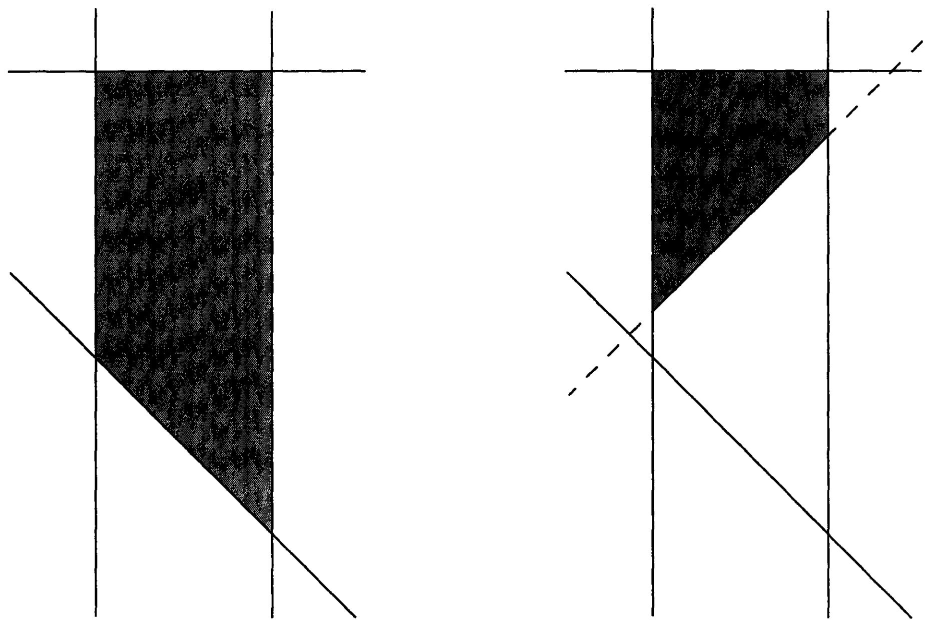

Figure 5-1: An example of a non-binding constraint becoming a binding one upon addition of another constraint. In both drawings, the lines represent constraints, and the shaded regions represent the feasible regions. The objective direction is the downward direction. Note that in the drawing on the left, the leftmost vertical line represents a non-binding constraint. In the drawing on the right, a single constraint, represented by the dashed line, has been added. The previously non-binding constraint becomes binding in this case.

two regions of the simplex. At first, this may seem computationally infeasible, as the set of all such cuts is exponential in size. However, we were able to bypass this difficulty with a clever application of max-flow/min-cut duality. Rather than list the exponentially many cuts separating the two terminals of each region, we wrote flow variables for pairs of adjacent vertices, and equated the flow between terminals to the value of the cut for that region.

In order to give the explicit linear program, we introduce some notation. Let $P$ denote the set of pair-isolating slices. Note that $P$ can be taken to be finite, since we only need to consider slices which induce different cuts in the finite point set $V$. Each plane $\pi \in P$ induces a cut $E_\pi \subseteq V \times V$, and partitions $V$ into two point sets, $A_\pi$ and $B_\pi$. Let $a_\pi^{(1)}$ and $a_\pi^{(2)}$ be the two terminals of $A$, and let $b_\pi^{(1)}$ and $b_\pi^{(2)}$ be the two terminals of $B$. We may then explicitly write the linear program as follows:

$$\max \lambda \;\; s.t.$$

$$\sum_s w_s \cdot |s| = 1$$

$$\forall \pi \in P : \begin{cases} \{f_u^{(\pi)} v\} \text{ a flow from } a_\pi^{(1)} \text{ to } a_\pi^{(2)} \text{ through } A_\pi \\ \{g_u^{(\pi)} v\} \text{ a flow from } b_\pi^{(1)} \text{ to } b_\pi^{(2)} \text{ through } B_\pi \\ f^{(\pi)}(a_\pi^{(1)}, a_\pi^{(2)}) + g^{(\pi)}(b_\pi^{(1)}, b_\pi^{(2)}) + \sum_{s \in E_\pi} w_s \geq \lambda \end{cases} \qquad \text{(FLOW-LP)}$$

Finally, for an additional touch of generality, we replaced $P$, the set of all planes induced by pair-isolating slices, with a more general class of pair-isolating geometric objects[1]. As alternatives, we considered $P^*$, the set of *all* planes that separate pairs of vertices. We also considered $W$, the set of all *wedge separators*. We define a wedge separator to be the non-planar geometric object produced by the following procedure:

1. Choose a point $p$ in the interior of the simplex.

2. For each face of the simplex, choose a projection of $p$ onto that face, taken along a vector parallel to an edge of the simplex. Note that there are three possible

[1]Recall that a pair-isolating slice is taken by using a plane parallel to two edges of the simplex.

projection points for each face.

3. Label the four projection points as $p_1$, $p_2$, $p_3$, and $p_4$ in some order.

4. Return the set $\angle p_1pp_2 \cup \angle p_2pp_3 \cup \angle p_3pp_4 \cup \angle p_4pp_1$, where $\angle abc$ denotes the set of all points contained in the planar region enclosed by the rays $\overrightarrow{ba}$ and $\overrightarrow{bc}$. Such a set of points is called a *wedge*; thus, a wedge-separator is the union of four wedges. When $\overrightarrow{ba}$ and $\overrightarrow{bc}$ are antiparallel, $\angle abc$ is undefined. We ignore such cases.

Figure 5-2 shows a graphical representation of a wedge-separator.

One can see that $W$ is a superset of $P$: a wedge separator generated by taking all projections along vectors parallel to either of two opposite edges of the simplex is a pair-isolating slice. The idea of wedge separators came from Karger et al. Their optimal cutting scheme for $k = 3$ uses what they refer to as *ball cuts*, generated by choosing an internal point and by taking side-parallel projections of that point to the three sides. A wedge separator seems to be a natural extension of this idea.

The computational complexity of FLOW-LP can be analyzed as follows. Let $n = |V|$, so that $n = \theta(d^3)$. There are then $O(M \cdot n)$ variables, where $M$ is the number of pair-isolating objects under consideration. The factor of $n$ is present to account for the flow variables. If we consider the set of all planes as pair isolators, this gives $M = O(n^3) = O(d^9)$, since each plane can be identified by three non-collinear points of the discretization grid. If we consider the set of all wedge separators, this gives $M = O(n) = O(d^3)$.

## 5.4 Results and conjectures

The discretization level we were able to use depended on whether or not we worked with the max-flow/min-cut paradigm, and by the set of pair-isolating objects we considered. When using the set of all planes and the set of all wedge separators in the max-flow/min-cut paradigm, we were able to achieve a discretization level of $d = 8$.

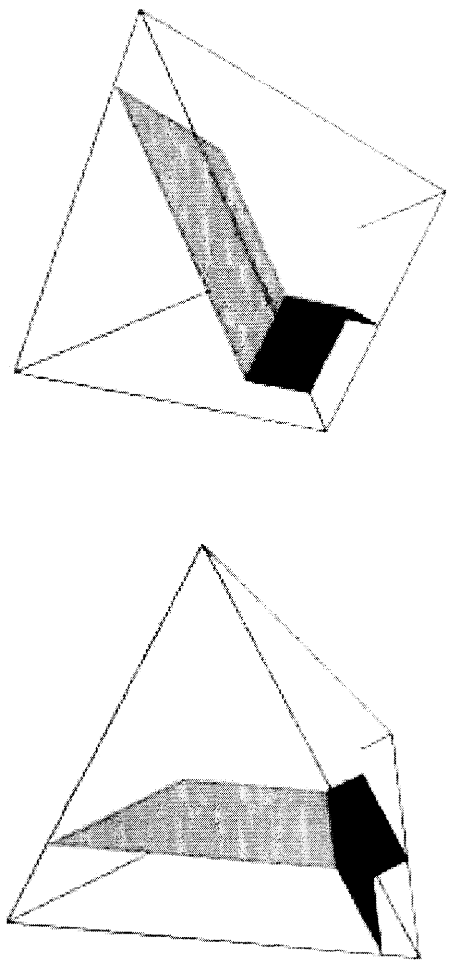

Figure 5-2: A graphical representation of a wedge separator, shown from two different angles. The four colored planar regions represent the four wedges. The center point $p$ is the point at which the four colored regions meet. In this particular example, the green and yellow wedges lie in the same plane (in a black and white version, these should be the two lightest colors).

Our results can be stated very succinctly. At no discretization level did *any* type of pair-isolating cut offer any improvement over sparcs. This leads us to the following conjecture:

**Conjecture 8.** *Pair-isolating cuts are not a necessary component of an optimal cutting scheme for the $k = 4$ case.*

As noted, our reasons for inferring this conjecture are based on a doubly-holed argument. We outline the argument here. If pair-isolating cuts are necessary for an optimal cutting scheme, then those cuts will be binding constraints of DUAL-$\infty$, the continuous version of LP-LB. This is implied by complementary slackness between PRI-$\infty$ and DUAL-$\infty$. Then, we may think of LP-LB as being generated by adding additional constraints to DUAL-$\infty$. These constraints force some of the edge weights to be zero and force others to be equal to each other. Our argument assumes that for sufficiently high discretization levels, the addition of these constraints preserve the bindingness of the binding constraints of DUAL-$\infty$. This is the first hole of our argument. It is not generally true that adding constraints to a linear program preserves the bindingness of others. However, in this context, there is reason to believe this should be the case for sufficiently high discretization levels, since the objective value of LP-LB approaches the objective value of DUAL-$\infty$ as the discretization level approaches $\infty$. This fact was proven by Karger et al. The second hole of our argument is independent of the first. Even if pair-isolating cuts correspond to binding constraints in LP-LB, those constraints may not be binding with respect to a proper subset (sparcs plus pair-isolating cuts) of all the constraints. This makes the opposite assumption of the first hole: that the addition of constraints to a linear program preserves the non-bindingness of others.

A stronger conjecture, based on our inability to conceive of other "natural" cuts for the $k = 4$ case besides sparcs and pair-isolating cuts, leads us to a more speculative conjecture:

**Conjecture 9.** *Sparcs are sufficient to describe the optimal cutting scheme for the $k = 4$ case.*

If Conjecture 9 is true, then sparcs are sufficient to describe the optimal cutting scheme for both $k = 3$ and 4. It is thus natural to conjecture the following:

**Conjecture 10.** *Sparcs are sufficient to describe the optimal cutting scheme for all $k$.*

# Chapter 6

# Conclusion

In this project, we have studied the integrality gap of the geometric relaxation given by Călinescu, Karloff, and Rabani for the minimum multiway cut problem. We have applied a stronger analysis to the computational experiments formulated by Karger, Klein, Stein, Thorup, and Young, to derive stronger upper bounds on the integrality gap. We have also devised an optimal minimum multiway cut branch-and-bound based algorithm, and have used this in a column generation paradigm to derive stronger lower bounds on the integrality gap.

The ultimate goal of researching the Călinescu et al. relaxation is to determine the optimal cutting scheme, as Karger et al. were able to do for the $k = 3$ case. This goal has eluded us. However, we believe that our computational experiments may shed some light on the nature of the optimal cutting scheme. In particular, our experimental results seem to indicate that an optimal cutting scheme for the $k = 4$ case may be expressible without pair-isolating cuts. Our inability to conceive of other "natural" cuts for the $k = 4$ case besides sparcs and pair-isolating cuts leads us to conjecture that the optimal cutting scheme for the $k = 4$ case may in fact be purely expressible as a probability distribution over sparcs. This conjecture is directly contrary to the conjecture of Mosk-Aoyama.

## 6.1 Future work

There are several natural directions in which to pursue the further study of the Călinescu et al. geometric relaxation. The ultimate goal, as noted, is to determine the optimal cutting scheme exactly. This can be thought of as two separate open problems: one is to construct an optimal cutting scheme, and the other is to construct a worst-case graph. It is unclear how one could recognize either as optimal without the other. If sparcs are indeed sufficient to construct the optimal cutting scheme for $k > 3$, it seems that careful examination of discretized solutions, combined with some brilliant pattern recognition, should lead to an analytical construction of an optimal cutting scheme. Despite our numerous efforts, this approach failed us, but we believe this to be the most promising line of research.

If, on the other hand, our conjecture is incorrect, and sparcs are *not* sufficient, the search for an optimal cutting schene becomes difficult. One must conceive of other classes of cuts, and one must either design computational experiments that incorporate these cuts or reason about the segment densities induced by these cuts analytically. Even for a relatively simple class of cuts like the pair-isolating cuts, we have discussed the difficulty of the designing tightly constrainted computational experiments in Section 5.1, and the resisted efforts of Mosk-Aoyama in his thesis seem to indicate the difficulty of making advances on the analytical side.

As for the problem of finding a worst case graph, our efforts have produced a stronger lower bound for various values of $k > 3$, but it will likely be difficult to extend our approach to find a tight matching lower bound as the efforts of Karger et al. did for the $k = 3$ case. The reason is that our procedure for determining worst-case graphs depends on a procedure that runs in time exponential in the graph size. We have utilized heuristics to ease the computational difficulty, but an exponential running time cannot be easily hidden. As a result, our discretization is too coarse, leaving us with insufficient data to recognize a pattern. Furthermore, even if a pattern could be found, it is not clear how one can analytically establish a lower bound on the graph's minimum multiway cut, as we cannot rely on planarity as Karger et al.

did for the $k = 3$ case. We made numerous attempts to generalize Karger et al.'s lower bound proof for $k > 3$, but none were successful.

# Bibliography

[1] M. Adams and V. Guillemin. *Measure Theory and Probability*, 1996. Birkhäuser Boston.

[2] Y. Boykov, O. Veksler, and R. Zabih, 1998. Markov Random Fields with Efficient Approximations. In *Proceedings of IEEE conference on "Computer Vision and Pattern Recognition"*, pages 648-655, 1998.

[3] G. Călinescu, H. Karloff, and Y. Rabani. An improved approximation algorithm for multiway cut. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, pages 48-52, 1998.

[4] E. Dahlhaus, D. S. Johnson, C. H. Papadimitriou, P. D. Seymour, and M. Yannakakis. The complexity of multiterminal cuts. In *SIAM Journal on Computing*, 23 (4): 864-894, 1994.

[5] J. Edmonds, R. Karp. Theoretical improvements in algorithmic efficiency for network flow problems. In *Journal of the ACM*, pages 248-264, 1972.

[6] A. Freund and H. Karloff. A lower bound of $8/(7 + \frac{1}{k-1})$ on the integrality ratio of the Călinescu-Karloff-Rabani relaxation for multiway cut. In *Information Processing Letters*, 75 (1-2): 43-50, 2000.

[7] P. C. Gilmore and R. E. Gomory. A linear programming approach to the cutting stock problem. In *Operations Research 9*, pages 848-859, 1961.

[8] D. R. Karger, P. Klein, C. Stein, M. Thorup, and N. E. Young. Rounding algorithms for a geometric embedding of minimum multiway cut. In *Proceedings of the*

*Thirty-First Annual ACM Symposium on Theory of Computing*, pages 668-678, 1999.

[9] M. Lübbecke and J. Desrosiers. Selected topics in column generation. In *Operations Research 53*, pages 1007-1023, 2005.

[10] G. Nicosia, A. Pacifici. Exact algorithms for a discrete metric labeling problem. In *Proceedings del CTW04 Workshop on Graphs and Combinatorial Optimization*, 2004.

[11] H. S. Stone. Multiprocessor scheduling with the aid of network flow algorithms. In *IEEE Transactions on Software Engineering*, 1977.