Approximation Algorithms for Hard Cut Problems

Research Thesis

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Computer Science

Julia Chuzhoy

Submitted to the Senate of
the Technion — Israel Institute of Technology

Elul 5760 Haifa September 2000

The research thesis was done under the supervision of Dr. Yuval Rabani in the faculty of computer science

I would like to thank Yuval Rabani for his devoted guidance throughout the course of this research

I would like to thank my husband Leonid for his support and encouragement

The generous financial help of the Technion is gratefully acknowledged

Contents

List of Figures

Abstract

The uniform labeling problem is defined in the following way: We are given an undirected graph $G = (V, E)$ with a set $T = \{t_1, \ldots t_k\} \subset V$ of special vertices called terminals, and non-negative weights w_e, for every $e \in E$. Additionally, for every $v \in V$ and for every $t \in T$, the assignment cost $c(v, t)$ is specified. We need to find an assignment function $f : V \setminus T \to T$ which assigns each non-terminal vertex to some terminal. Each such assignment function has a cost, and we need to find the assignment function of minimum cost. A cost of an assignment function consists of two parts:

1. Vertex assignment cost, which is $\Sigma_{v \in V} c(v, f(v))$.
2. Edge separation cost: $\Sigma_{\substack{(u,v) \in E \\ f(u) \neq f(v)}} w_{(u,v)}$

The total cost of an assignment function is the sum of the vertex assignment cost and the edge stretch cost.

The problem can be viewed as a labeling problem, where the non-terminal vertices represent the objects we want to label and the terminals represent the labels. The assignment function f actually assigns to each object some label. The vertex assignment costs $c(v, t)$ reflect our initial estimation of how likely it is that the object corresponding to v should get a label corresponding to t. Finally, there is an edge between two vertices if it is likely that the two corresponding objects should get the same label, and the higher the edge weight is, the more likely it is that the two objects should get the same label.

For the special case of $k = 2$ the problem can be easily transformed to the s-t cut problem, and can be solved in polynomial time. For all $k \geq 3$, the problem is NP-hard and also Max SNP-hard. Therefore, unless $P = NP$, there is no polynomial time approximation scheme for the problem. Kleinberg and Tardos showed a 2-approximation algorithm for the problem. In this work we concentrate on two special cases of the problem. For the special case of $k = 3$, we show a factor $\frac{4}{3}$ approximation algorithm, which matches the integrality gap of the relaxation we use. For $k = 4$, we show a factor $\frac{3e^{\frac{1}{3}}-1}{2} \approx 1.5934187$ approximation algorithm, while the integrality gap of the relaxation we use is at least 1.5.

Notations and Abbreviations

Δ_k — the $(k-1)$-simplex, which is a convex polytope in $\mathbb{R}^k$, given by: $\Delta_k = \{x \in \mathbb{R}^k | x \geq 0 \text{ and } \Sigma_{i=1}^k x_i = 1\}$

$\|x\|$ — For $x \in \mathbb{R}^k$, $\|x\|$ is its L_1 norm: $\|x\| = \Sigma_{i=1}^k |x_i|$.

e^i — a unit vector with $e_i^i = 1$ and $e_j^i = 0$ for all $j \neq i$.

$|e|$ — for a graph edge $e = (u, v)$, $|e| = \frac{1}{2}\|x^u - x^v\|$

Chapter 1

Introduction

1.1 Introduction

The *uniform labeling problem* is defined in the following way: we are given an undirected graph $G = (V, E)$, with nonnegative edge weights w_e for each $e \in E$ and a set $T = \{t_1 \cdots t_k\} \subseteq V$ of special vertices called terminals. For each vertex $v \in V \setminus T$ and for each terminal $t \in T$, $c(v, t) \geq 0$ denotes the cost of assigning v to t. We need to find a function $f : V \setminus T \to T$, which assigns each vertex to some terminal. The cost of an assignment is divided into two parts:

1. Vertex assignment cost: the cost we pay for the assignment of the vertices, which is $\Sigma_{v \in V} c(v, f(v))$
2. Edge separation cost: for each edge whose endpoints are assigned to different terminals we pay the weight of the edge, the total being

   $$\Sigma_{\substack{(u,v) \in E \\ f(u) \neq f(v)}} w_{(u,v)}$$

The total cost of an assignment is the sum of the vertex assignment cost and the edge stretch cost. We are looking for the assignment with minimum assignment cost.

Intuitively, the non-terminal vertices of the graph can be viewed as objects, and the terminals as labels. The assignment of the non-terminal vertices to the terminals can be viewed as labeling the objects. Edges between vertices denote that the corresponding objects are related, and the bigger the edge cost is, the more likely it is that the corresponding objects should get the same label. Finally, the assignment costs show how likely it is that a certain object gets a certain label. The higher the cost $c(v, t_i)$ is, the less likely it is that the object v should get the label t_i.

For $k = 2$, the problem can be easily transformed to the undirected version of the s-t cut problem of Ford and Fulkerson [4]. The transformation is as follows: We denote by s and t the two terminals. For each vertex $v \in V$ we add two new edges to the graph: (v, t) with the weight $w_{(v,t)} = c(v, s)$ and (v, s) with the weight $w_{(v,s)} = c(v, t)$. The two new edges will "pay" the assignment cost of the vertex v. For example, if v is assigned to t, the edge (v, s) is cut, and the stretch cost this edge pays is exactly $w_{(v,s)} = c(v, t)$ — the cost of assigning v to t.

For $k \geq 3$ the problem is NP-hard and also Max SNP-hard. Therefore, unless $P = NP$, there is no polynomial time approximation scheme for this problem. Kleinberg and Tardos [8] give a polynomial time 2-approximation algorithm, and this is the best approximation ratio previously known for this problem.

We considered two special cases of this problem: For $k = 3$, we show a $\frac{4}{3}$-approximation algorithm, which matches the integrality gap of the relaxation we use. For $k = 4$, we show a $\frac{3e^{\frac{1}{3}}-1}{2} \approx 1.5934187$-approximation algorithm, while the integrality gap of the relaxation we use is at least 1.5.

1.2 Related Problems

1.2.1 Metric labeling problem

The *Metric labeling problem* is a generalization of the uniform labeling problem, where a metric distance function $d(\cdot, \cdot)$ on the terminals is defined. The

vertex assignment cost remains the same, and the edge stretch cost becomes $\Sigma_{(u,v)\in E} w_{(u,v)} \cdot d(f(u), f(v))$. Intuitively, the distances between the terminals show how far apart the corresponding labels are. Kleinberg and Tardos [8] gave a factor $O(\log k \log\log k)$–approximation algorithm for this problem (k is the number of the terminals).

Gupta and Tardos [5] considered a special case of the metric labeling problem, where the metric is the truncated linear metric. They give a 4-approximation algorithm for this problem.

1.2.2 Multiway cut

In the *multiway cut* problem we are given an undirected graph G with non-negative edge weights, and a set of terminals. We need to assign each vertex to some terminal, so as to minimize the total weight of the edges whose endpoints are assigned to different terminals.

The problem can be viewed as a special case of the uniform labeling problem, with 0 assignment costs. Dahlhaus, Johnson, Papadimitriou, Seymour and Yannakakis [3] proved that multiway cut is Max SNP-hard. Therefore, unless P=NP, there is no polynomial-time approximation scheme for multiway cut, and hence for the uniform labeling problem as well. The first constant factor approximation algorithm for this problem is due to Dahlhaus, Johnson, Papadimitriou, Seymour and Yannakakis [3], who gave a $2 - \frac{2}{k}$-approximation algorithm for this problem (k is the number of the terminals). The algorithm is based on computing the minimum isolating cut C_i for each terminal t_i (a cut that isolates t_i from all the other terminals). Let C_m denote the maximum weight cut between $C_1, \ldots, C_k$. The multiway cut is the union of all the isolating cuts C_i, except for C_m. Recently Călinescu, Karloff and Rabani [1] found a $(\frac{3}{2} - \frac{1}{k})$ approximation algorithm, using a new geometric relaxation of multiway cut. This result was improved by Karger, Klein, Stein, Thorup and Young [6], which give a $(1.3438 - \epsilon_k)$ approximation algorithm for each k, using the same relaxation. For $k = 3$, Cunningham and Tang [2] and Karger et al. [6] give a $\frac{12}{11}$ approximation algorithm using the same relaxation and also prove that $\frac{12}{11}$ is the integrality gap for this relaxation. A similar relaxation was used by Kleinberg and Tardos [8] for their 2-approximation

algorithm for the uniform labeling problem. We use the same relaxation to achieve the approximation algorithms for the uniform labeling problem.

Dahlhaus, Johnson, Papadimitriou, Seymour and Yannakakis studied in [3] the special case of the multiway cut problem where the graph is a planar graph. They showed polynomial time algorithms for this problem for all fixed values of k. For k part of an input, they showed that the problem is NP-hard.

1.3 Work Overview and Main Methods

1.3.1 Integer Programming and Linear Programming

A general form of an *integer program* is as follows:

$$\begin{aligned} \textit{minimize} \quad & cx \\ s.t. \quad & Ax \geq b && (1.1) \\ & x_i \in \{0,1\} \quad \forall i \in \{1,\ldots,n\} && (1.2) \end{aligned}$$

Where $c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$ and $A \in \mathbb{R}^{m \times n}$. Finding an optimal solution to an integer program is an NP-hard problem.

A *linear program* (in canonical form) is the following:

$$\begin{aligned} \textit{minimize} \quad & cx \\ s.t. \quad & Ax \geq b, && (1.3) \end{aligned}$$

where $c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$ and $A \in \mathbb{R}^{m \times n}$. Linear programs can be solved in polynomial time, for example, using Karmarkar's algorithm [7].

A widely used technique (which we also use in this work) for obtaining a factor α approximation algorithms for combinatorial problems, is as follows: First,

we express the combinatorial problem as an integer program. The entries of A, b and c and the parameters n and m may depend on the instance of the combinatorial problem. We denote the integer program by (IP). By relaxing the integrality constraints (1.2), we get a linear program (LP), which is a relaxation of (IP). Note that given any instance S of the combinatorial problem, we can solve the corresponding instance of (LP) in polynomial time. The solution we obtain is called an optimal fractional solution, and it does not necessarily meet the integrality constraints (1.2) of the integer program. To obtain a solution to the integer program, we must round the optimal fractional solution.

So we try to construct a rounding algorithm A, which, given a fractional solution of cost c outputs an integer solution of cost $\leq \alpha c$. If we succeed in constructing such an algorithm, we can compute α-approximation for any instance S of the combinatorial problem, by first solving the corresponding instance of the linear program, and then applying A to the optimal fractional solution. Given any problem instance S, we denote by $OPT_{IP(S)}$ and $OPT_{LP(S)}$ the costs of the optimal integer and the optimal fractional solutions for S, respectively, and we denote by $C_{IP(S)}$ the cost of the solution A outputs when it is given an optimal fractional solution for S as input. Since any feasible solution of the integer program is also a feasible solution of the linear program, we know that $OPT_{LP(S)} \leq OPT_{IP(S)}$. Therefore, $C_{IP(S)} \leq \alpha \cdot OPT_{LP(S)} \leq \alpha \cdot OPT_{IP(S)}$, and so we get a factor α approximation.

A general way to compute a lower bound on the approximation factor which can be achieved using the technique described above is to compute a lower bound on the integrality gap of the relaxation.

Suppose we have an integer program (IP) and its linear relaxation (LP) for some combinatorial problem. Let S be some instance of this problem. Let $OPT_{LP(S)}$ be the cost of the optimal fractional solution for this instance and $OPT_{IP(S)}$ be the cost of the optimal integer solution for this instance. Then the *integrality gap* of the relaxation is the maximum of $\frac{OPT_{IP(S)}}{OPT_{LP(S)}}$ over all possible instances S of the problem. If we can show a problem instance S with $\frac{OPT_{IP(S)}}{OPT_{LP(S)}} \geq \beta$, then we cannot achieve approximation factor $< \beta$ using the technique described above, since no rounding algorithm A which achieves

factor $< \beta$ exist: No rounding algorithm A, given the optimal fractional solution to S of cost $OPT_{LP(S)}$ can output an integer solution to S of cost $< \beta OPT_{LP(S)}$, since $OPT_{IP(S)} \geq \beta OPT_{LP(S)}$.

1.3.2 Work Overview

In chapter 2 we define the restricted multiway cut problem. This problem is a special case of the uniform labeling problem. We write an integer program for this problem and its relaxation — a linear program. We show that the integrality gap of the relaxation is at least $2 - \frac{2}{k}$ (k is the number of the terminals). In chapter 3 we show two rounding algorithms for this linear program. The first algorithm works for the special case of $k = 3$ and achieves an approximation factor of $\frac{4}{3}$, which matches the integrality gap for $k = 3$. The second algorithm works for the special case of $k = 4$ and achieves approximation factor $\frac{3e^{\frac{1}{3}}-1}{2} \approx 1.5934187$, while the integrality gap for $k = 4$ is at least 1.5.

In chapter 4 we write an integer program and its linear relaxation for the uniform labeling problem and show that the two rounding algorithms for the restricted multiway cut problem can be used to round fractional solutions of the uniform labeling problem (for the same values of k), achieving the same approximation ratios.

Chapter 2

The Restricted Multiway Cut Problem

We next define the *restricted multiway cut problem*, which is a special case of the uniform labeling problem.

In the next chapter we show approximation algorithms for this problem, which is simpler and easier to deal with than the uniform labeling problem. In chapter 4 we show that these algorithms can also be used as approximation algorithms for the uniform labeling problem, achieving the same approximation ratios.

2.1 Problem Definition

In the restricted multiway cut problem, we are given an undirected graph, $G = (V, E)$ with non-negative edge weights w_e for each $(u, v) \in E$, and a set of special vertices $T = \{t_1 \dots t_k\} \subseteq V$ called terminals. For each vertex $v \in V \setminus T$ a list $l_v \subseteq T$ of possible assignments to the terminals is specified. We need to assign each non-terminal graph vertex $v \in V$ to some terminal $t \in l_v$. The cost of the assignment is the sum of the weights of edges, whose endpoints are assigned to different terminals.

This problem can be viewed as a special case of the uniform labeling problem, where the assignment costs are restricted only to 0 and M, where M is a large integer ($M > \Sigma_{e \in E} w_e$ is enough): for each $v \in V$, $c(v,t) = 0$ if $t \in l_v$, and $c(v,t) = M$ otherwise. Indeed if there is a feasible solution to the restricted multiway cut problem, its cost must be smaller than M. So the optimal solution to the problem we obtained after the transformation must also have a cost smaller than M and thus it is a feasible solution to the restricted multiway cut problem.

2.2 Linear Program for the Restricted Multiway Cut Problem

Notation: We use Δ_k to denote the $(k-1)$-simplex, which is a convex polytope in $\mathbb{R}^k$, given by: $\Delta_k = \{x \in \mathbb{R}^k | x \geq 0 \; and \; \Sigma_{i=1}^k x_i = 1\}$

For $x \in \mathbb{R}^k$, $\|x\|$ is its L_1 norm: $\|x\| = \Sigma_{i=1}^k |x_i|$.

Finally, we use e^i for a unit vector with $e_i^i = 1$ and $e_j^i = 0$ for all $j \neq i$.

The linear program: We represent each graph node v by a vector $x^v \in \mathbb{R}^k$. The vector x^v has $x_i^v = 1$ if v is assigned to the terminal t_i. Otherwise, $x_i^v = 0$. This can be viewed as placing all the terminals on the vertices of a $(k-1)$-simplex, and then assigning the graph nodes to the terminals by placing them on the corresponding simplex vertices. We also introduce a vector variable $z^{(u,v)} \in \mathbb{R}^k$ for each $(u,v) \in E$, and its value is $z^{(u,v)} = |x^u - x^v|$.

$$(MIP) \quad Min \; \tfrac{1}{2} \Sigma_{(u,v) \in E} w_{(u,v)} \cdot (\Sigma_{i=1}^k z_i^{(u,v)})$$

$$\begin{aligned} s.t. \quad & x^{t_i} = e^i && i = 1, \ldots, k && (2.1) \\ & \Sigma_{i=1}^k x_i^v = 1 && \forall v \in V && (2.2) \\ & x_i^v = 0 && \forall v \in V, \; \forall i \; : \; t_i \notin l_v && (2.3) \end{aligned}$$

$$z_i^{(u,v)} \geq x_i^v - x_i^u \quad \forall (u,v) \in E,\ i = 1,\ldots,k \tag{2.4}$$
$$z_i^{(u,v)} \geq x_i^u - x_i^v \quad \forall (u,v) \in E,\ i = 1,\ldots,k \tag{2.5}$$
$$x_i^v \in \{0,1\} \quad \forall v \in V,\ i = 1,\ldots,k \tag{2.6}$$

The constraints (2.4), (2.5) ensure that $z_i^{(u,v)} \geq |x_i^u - x_i^v|$. Since in the objective function we want to minimize $z_i^{(u,v)}$, it is clear that in the optimal solution, for each $(u,v) \in E$ and $i \in \{1,\ldots k\}$, $z_i^{(u,v)} = |x_i^u - x_i^v|$ holds. Thus the objective function is actually:

$$Min\ \frac{1}{2}\Sigma_{(u,v)\in E} w_{(u,v)} \cdot \|x^u - x^v\|.$$

In this paper we define the length of an edge $e = (u,v)$ to be half the L_1 distance in between x^u and x^v: $|e| = \frac{1}{2}\|x^u - x^v\|$, and thus the objective function of the integer program is:

$$Min\ \frac{1}{2}\Sigma_{e\in E} w_e \cdot |e|.$$

Clearly, any feasible solution for (MIP) places all the graph nodes in the simplex vertices, which can be viewed as assigning the graph nodes to the corresponding terminals. The constraints (2.3) ensure that each vertex v can be placed only on the simplex vertices which correspond to the terminals from l_v. For each $(u,v) \in E$, $\frac{1}{2}\|x^u - x^v\| = 0$ if u,v are assigned to the same simplex vertex, and $\frac{1}{2}\|x^u - x^v\| = 1$ otherwise. Thus, we pay $w_{u,v}$ for an edge (u,v) if u and v are assigned to different terminals. Thus, any feasible solution to (MIP) gives us a solution to the restricted multiway cut problem of the same or smaller cost. (The cost may be smaller if for some $(u,v) \in E$ and $i \in \{1,\ldots,k\}$, the value $z_i^{(u,v)}$ is $> |x_i^u - x_i^v|$.)

Note also that given any solution to the restricted multiway cut problem, we can construct a feasible solution of the same cost to the (MIP), by placing all the graph nodes on the simplex vertices together with the terminals they are assigned to (i.e. we set $x^v = e^i$, where i is the index of the terminal v is assigned to), and setting $z^{(u,v)} = |x^u - x^v|$ for each $(u,v) \in E$. Thus, (MIP) and the restricted multiway cut problem are equivalent.

We denote by (MLP) the linear program obtained from (MIP) by replacing the integrality constraints (2.6) with

$$x_i^v \geq 0 \quad \forall v \in V,\ i = 1, \ldots, k \tag{2.7}$$

This linear program can be solved in polynomial time, giving us a fractional solution to the restricted multiway cut problem, where each graph node v is placed on the simplex, on the facet spanned by the terminals from l_v. We will try to round this solution by placing all the nodes on simplex vertices, while trying to increase the cost of the solution by as little as possible. Note that we must make sure that if some vertex v is on the facet spanned by some terminals, it will be assigned to one of these terminals. For example, if $x_i^v = 0$, then v will not be assigned to t_i (because it is possible that $t_i \notin l_v$).

2.3 Integrality Gap of the Relaxation

Kleinberg and Tardos proved in [8] that the integrality gap of the relaxation, which they used for the uniform labeling problem, and which we use later in this work, is $\geq 2 - \frac{2}{k}$. The proof of the integrality gap of our relaxation for the restricted multiway cut problem easily follows from the proof given in [8].

We show that for each k, the integrality gap of the relaxation (MLP) is $\geq 2 - \frac{2}{k}$. To show this, we first show an instance of the restricted multiway cut problem and give a fractional solution for this instance of cost $\frac{k}{2}$. We then show that the cost of the optimal integer solution for this instance is $k - 1$. Since the cost of the optimal fractional solution for this instance is $\leq \frac{k}{2}$, the integrality gap is $\geq \frac{k-1}{\frac{k}{2}} = 2 - \frac{2}{k}$.

Consider the following instance of the restricted multiway cut problem: $G = (V, E)$, $V = \{v_1, \ldots, v_k\} \cup \{t_1, \ldots, t_k\}$, $T = \{t_1, \ldots, t_k\}$, $E = \{(v_i, v_j) | 1 \leq i, j \leq k,\ i \neq j\}$. For each $e \in E$, $w_e = 1$. For each $i \in \{1, \ldots, k\}$, $l_{v_i} = T \backslash \{t_i\}$

The fractional solution we propose is as follows: For each $i \in \{1, \ldots, k\}$, the i-th coordinate of x^v is 0, and all the other coordinates are $\frac{1}{k-1}$. There are

$\binom{k}{2}$ edges in the graph, each one of length $\frac{1}{k-1}$ and weight 1. Thus the cost of the fractional solution is $\binom{k}{2} \cdot \frac{1}{k-1} = \frac{k}{2}$.

In an integer solution, we cannot assign all the vertices to one terminal, since for each $i \in \{1, \ldots, k\}$, $t_i \notin l_{v_i}$. So the best thing we can do is to assign all the vertices to two terminals. It is not hard to see that the optimal integer solution is achieved, for example, by assigning $v_2 \ldots v_k$ to t_1, and v_1 to t_2. The cost of this solution is $k-1$, and so the integrality gap is $\geq \frac{k-1}{\frac{k}{2}} = 2 - \frac{2}{k}$.

$\binom{k}{2}$ edges in the graph, each one of length $\frac{1}{k-1}$ and weight 1. Thus the cost of the fractional solution is $\binom{k}{2} \cdot \frac{1}{k-1} = \frac{k}{2}$.

In an integer solution, we cannot assign all the vertices to one terminal, since for each $i \in \{1, \ldots, k\}$, $t_i \notin l_{v_i}$. So the best thing we can do is to assign all the vertices to two terminals. It is not hard to see that the optimal integer solution is achieved, for example, by assigning $v_2 \ldots v_k$ to t_1, and v_1 to t_2. The cost of this solution is $k-1$, and so the integrality gap is $\geq \frac{k-1}{\frac{k}{2}} = 2 - \frac{2}{k}$.

Chapter 3

Upper Bounds for the Restricted Multiway Cut Problem

We are looking for algorithms of the following form:

Alg.:

1. Choose a random permutation σ of $\{1 \dots k\}$, and let σ_i denote the i-th number in this permutation.
2. For $i = 1 \dots k-1$
   - Choose $\rho_i \in (0,1)$ with some distribution
   - Assign to t_{σ_i} all the vertices $v \in V$ with $x^v_{\sigma_i} \geq \rho_i$ that were not assigned previously.
3. Assign to t_{σ_k} all the vertices that are not assigned yet.

In order for such a solution to work, we need that $\Sigma_{i=1}^{k-1} \rho_i \leq 1$. Otherwise it is possible that there will be some vertex v on the facet spanned by $T \setminus \{t_{\sigma_k}\}$ (with $x^v_{\sigma_k} = 0$), which will not be assigned in step 2, and thus will be assigned to t_{σ_k}. This can give an infeasible solution, because it is possible that $t_{\sigma_k} \notin l_v$.

Lemma 1 *Any solution obtained by any rounding algorithm of the form Alg., where* $\Sigma_{i=1}^{k-1}\rho_i \leq 1$ *is a feasible solution for the restricted multiway cut problem.*

Proof:

We must make sure that if for some vertex v and for some $q \in \{1 \ldots k\}$, $x_q^v = 0$, then v is never assigned to t_q.

Let $v \in V$ be some vertex, and suppose $x_q^v = 0$ for some $q \in \{1 \ldots k\}$, and suppose $q = \sigma_j$, $j \in \{1 \ldots k\}$.

If $j \neq k$, v can only be assigned to t_q if $\rho_j \leq x_q^v = 0$, which is impossible since $\rho_j > 0$.

Now suppose that $j = k$. Then v can be assigned to t_{σ_k} only if v was not assigned to any of the $t_{\sigma_i} : i < k$. This means that for each $i < k$, $x_{\sigma_i}^v < \rho_i$, and thus $\Sigma_{i=1}^{k-1} x_{\sigma_i}^v < \Sigma_{i=1}^{k-1}\rho_i \leq 1$. This is impossible, because $\Sigma_{i=1}^{k-1} x_{\sigma_i}^v = 1$.
□

3.1 Some Useful Observations

The next lemma states that given a fractional solution to the restricted multiway cut problem, we can assume that each graph edge is parallel to some simplex edge, i.e., for each graph edge, its two endpoints differ in at most 2 coordinates. The lemma was proved in [1]. We include it here for the sake of completeness.

Lemma 2 *Given a fractional solution S to the restricted multiway cut problem of cost c, one can obtain from it another solution, S', to the restricted multiway cut problem, of cost $c' \leq c$, in which for each edge $(u,v) \in E$, x^u and x^v differ in at most two coordinates.*

Proof: While there exists an edge $(v,u) \in E$, such that x^u and x^v differ in more than two coordinates, we perform the following procedure:

Let q be the number of coordinates in which x^u and x^v differ, and let i be the coordinate in which the two nodes differ, which minimizes $d_i = |x_i^v - x_i^u|$. We assume without loss of generality, that $x_i^v < x_i^u$. Since $\Sigma_{i=1}^k x_i^u = \Sigma_{i=1}^k x_i^v = 1$, there exists an index j, such that $x_j^v > x_j^u$. Note that $d_j = x_j^v - x_j^u \geq d_i$.

We add a new vertex v' to the graph, with $l_{v'} = T$, and set $x^{v'}$ in the following way: for each $r \neq i, r \neq j$, $x_r^{v'} = x_r^v$. $x_i^{v'} = x_i^v + d_i = x_i^u$, and $x_j^{v'} = x_j^v - d_i$.

We remove the edge (v, u) from the graph and add instead two new edges (v, v') and (v', u), with $w_{(v,v')} = w_{(v',u)} = w_{(u,v)}$.

Note that x^v and $x^{v'}$ differ in two coordinates, so the number of "bad" edges did not increase. Note also that $x^{v'}$ and x^u differ in at most $q-1$ coordinates. So we need to repeat this procedure at most $(k-2)|E|$ times.

Finally, we show that the cost of the solution did not increase. The only change in the cost concerns the edge costs of (v, u), (v, v') and (v', u).

Before the transformation we paid $\frac{1}{2}\|x^u - x^v\| w_{(v,u)}$. After the transformation, we pay $\frac{1}{2} w_{(v,u)}(\|x^v - x^{v'}\| + \|x^{v'} - x^u\|)$. Note that $\|x^v - x^{v'}\| = 2d_i$.

$$
\begin{array}{lcl}
\|x^{v'} - x^u\| & = & \\
\|x^v - x^u\| & - & (|x_i^v - x_i^u| + |x_j^v - x_j^u|) + \\
 & + & (|x_i^u - x_i^u| + |x_j^v - d_i - x_j^u|) = \\
\|x^v - x^u\| & - & d_i - (x_j^v - x_j^u) + x_j^v - d_i - x_j^u = \\
\|x^v - x^u\| & - & 2d_i
\end{array}
$$

(Since we know that $x_j^v > x_j^u$ and $x_j^v - x_j^u \geq d_i$.)

Thus $\frac{1}{2} w_{(v,u)}(\|x^v - x^{v'}\| + \|x^{v'} - x^u\|) = \frac{1}{2} w_{(v,u)} \|x^v - x^u\|$, and the cost of the solution did not increase. □

Note that now, given an integer solution to S', we can easily transform it to an integer solution to S of the same or even smaller cost, by removing the additional vertices and edges we added during the transformation, and restoring the edges from the original graph.

Thus we can assume now that in a fractional solution to the restricted multiway cut problem all the graph edges are parallel to the simplex edges — a

simplifying assumption which we use in our algorithms.

Lemma 3 *Suppose we are given $\Delta > 0$ and a fractional solution S of (MLP) where the endpoints of each edge differ in at most two coordinates. We can obtain from it another solution, S', for which the following condition holds:*

For each edge $e = (u, v)$, for each $i \in \{1, \ldots, k\}$, either both x_i^v and x_i^u are $\geq \Delta$, or they are both $\leq \Delta$.

Proof: While there exists an edge which violates this condition, let $e = (u, v)$ be such an edge. We denote its length by ϵ. We assume, without loss of generality, that the edge's endpoints differ in the first two coordinates, and that $x_1^v > x_1^u$. Thus, $x_1^v - x_1^u = x_2^u - x_2^v = \epsilon$.

We can only have problems with the first two coordinates of the edge's endpoints, since all the other coordinates are equal in both endpoints.

We must take care of three cases:

1. $x_1^v > \Delta > x_1^u$, but x_2^v and x_2^u are both either $\geq \Delta$ or $\leq \Delta$.

   We add a new vertex v' to the graph, with $x_1^{v'} = \Delta$, $x_2^{v'} = x_2^v + x_1^v - \Delta$, and for each $2 < i \leq k$, $x_i^{v'} = x_i^v = x_i^u$. We set $l_{v'} = T$. We replace the edge (v, u) by two new edges: (v, v') and (v', u) with $w_{(v,v')} = w_{(v',u)} = w_{(v,u)}$. Note that the cost of the solution remains the same:

   $$\|x^v - x^{v'}\| = 2(x_1^v - \Delta)$$

   $$\begin{aligned} &\|x^{v'} - x^u\| = \\ &|\Delta - x_1^u| + |x_2^u - x_2^v - x_1^v + \Delta| = \\ &|\Delta - x_1^u| + |x_1^v - x_1^u - x_1^v + \Delta| = \\ &2(\Delta - x_1^u) \end{aligned}$$

   Thus,

$$\frac{1}{2}w_{(v,v')}\|x^v - x^{v'}\| + \frac{1}{2}w_{(v',u)}\|x^{v'} - x^u\| =$$
$$w_{(u,v)}(x_1^v - x_1^u) = \frac{1}{2}w_{(u,v)}\|x^v - x^u\|$$

Note that $x_1^v \geq x_1^{v'} = \Delta$, $x_1^u < x_1^{v'} = \Delta$. Also $x_2^v, x_2^u, x_2^{v'}$ are all either $\geq \Delta$ or $\leq \Delta$. This is because $x_2^v \leq x_2^{v'} \leq x_2^u$.

2. $x_2^u > \Delta > x_2^v$, but x_1^u and x_1^v are both either $\geq \Delta$ or $\leq \Delta$. In this case we can exchange the first coordinate with second, v with u and get exactly the first case. So we can proceed like in the previous case.

3. $x_1^v > \Delta > x_1^u$ and $x_2^u > \Delta > x_2^v$. In this case we can first "fix" the first coordinate, as in case 1. As the result we will get two new edges, (v,v') and (u,v'), with $x_1^v, x_1^{v'} \geq \Delta$, $x_1^u, x_1^{v'} \leq \Delta$, and $x_2^u \geq x_2^{v'} \geq x_2^v$. It is possible that either $x_2^u > \Delta > x_2^{v'}$ or $x_2^{v'} > \Delta > x_2^v$, but not both. In this case we have to fix the second coordinate of the problematic edge, like in case 2.

We must perform this procedure at most once for each edge.
□

Again, given an integer solution to S' we can obtain an integer solution to S of smaller or equal cost by removing all the additional edges and vertices we have added to the graph and restoring the original edges.

3.2 Upper Bound for $k = 3$

For $k = 3$ we show the upper bound of $\frac{4}{3}$, which matches the lower bound on the integrality gap.

The Rounding Algorithm: We use an algorithm of the form *Alg.* for $k = 3$.

The parameter ρ_1 is distributed uniformly at random in $(0, 1)$, and $\rho_2 = 1 - \rho_1$.

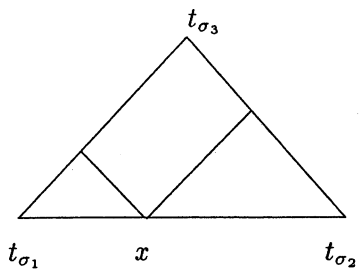

Figure 3.1: The rounding algorithm for $k = 3$

Note that $\rho_1 + \rho_2 = 1$, and thus, by Lemma 1, the solution is feasible.

The algorithm can be viewed geometrically in the following way: we choose a point on the boundary of the simplex uniformly at random. Suppose we have chosen a point x on one of the three simplex edges, we cut the triangle with two lines which pass through this point and are parallel to the other two simplex edges.

Analysis We show that for each edge $e = \{(u, v)\}$ of length ϵ, the probability that the edge is cut by the algorithm is $\frac{4}{3}\epsilon$. Thus the expected cost of the solution is $\Sigma_{(u,v)\in E}\frac{1}{2}w_{(u,v)}\|x^u - x^v\| \cdot \frac{4}{3}$, and so the approximation ratio of the algorithm is $\frac{4}{3}$.

Let $e = (u, v)$ be some edge of G, of length ϵ. We assume without loss of generality that $x_3^u = x_3^v$. Thus $|x_1^u - x_1^v| = |x_2^u - x_2^v| = \epsilon$. We consider all the possible permutations. For each we compute the probability over all values of ρ that the edge will be cut.

If the first two coordinates in the permutation are 1 and 2 (which happens with probability $\frac{1}{3}$), the edge is cut with probability $\leq 2\epsilon$.

If the first two coordinates are 1 and 3 (which also happens with probability $\frac{1}{3}$), the edge is cut with probability ϵ.

If the first two coordinates are 2 and 3, the case is symmetrical to the previous one.

The total probability that the edge will be cut during the algorithm is $\frac{1}{3}(2\epsilon + \epsilon + \epsilon) = \frac{4}{3}\epsilon$.

3.3 Upper Bound for $k = 4$

Our Computational Experiment We performed computational experiments similar to those performed by Karger et al. [6].

We wanted to use again an algorithm of the form Alg. So we were looking for the distribution of ρ_1, ρ_2, ρ_3, such that $\rho_1 + \rho_2 + \rho_3 \leq 1$, which would give us the best approximation ratio, using algorithm of the form Alg. We tried to get some idea how this distribution looks by solving a linear program.

First, we have built a mesh (with step d). The vertices of the mesh are:

$$V = \{(\frac{i}{d}, \frac{j}{d}, \frac{k}{d}, \frac{l}{d}) \mid \frac{i}{d} + \frac{j}{d} + \frac{k}{d} + \frac{l}{d} = 1,\ i, j, k, l \text{ are integers}\}$$

There is an edge between any two vertices which are at distance $\frac{1}{d}$ from one another.

Note now that we need not consider all the possible values of ρ_i ($i \in \{1, 2, 3\}$). It's enough to consider the values which are multiples of $\frac{1}{d}$.

So the variables of the linear program are all the triples (ρ_1, ρ_2, ρ_3), where ρ_1, ρ_2, ρ_3 are multiples of $\frac{1}{d}$, and $\rho_1 + \rho_2 + \rho_3 \leq 1$. The values of these variables correspond to the probability of choosing this triple as a solution. Additional variable is MC, which is the approximation factor of the solution.

We try to minimize MC, subject to the constraints that for each edge e, the probability that the edge is cut is less than or equal to $MC \cdot |e|$.

Clearly, the smaller the step ($\frac{1}{d}$) we choose, the bigger the linear program becomes. So we were able to solve the linear program only for $d \leq 18$. The best approximation ratio achieved using this way by the linear program with $d = 18$ was 1.54. It shows that there does not exist a rounding algorithm of the form presented above, which achieves approximation factor 1.5 (the

lower bound on the integrality gap for the relaxation we use). There was also a tendency of the solution for the linear program (the approximation factor) to become bigger as we increased d. So it is possible that factor 1.54 also cannot be achieved in this way.

The solution of the linear program gave us the idea of the distributions of ρ_1, ρ_2, ρ_3 in the rounding algorithm which we present here.

The Intuition: We use the algorithm of the form *Alg.* described above. So we only need to specify what are the distributions of ρ_1, ρ_2 and ρ_3. We use a parameter Δ, and we will set its value later. We denote by $l = 1 - 3\Delta$.

The values of ρ_1 and ρ_2 are distributed uniformly at random, between Δ and $1 - 2\Delta$, such that their sum is always $1 - \Delta$. This is achieved by choosing ρ_1 uniformly at random between Δ and $1 - 2\Delta$ and then setting $\rho_2 = 1 - \Delta - \rho_1$. The parameter ρ_3 is distributed between 0 and Δ. Note that $\rho_1 + \rho_2 + \rho_3 \leq \Delta + 1 - \Delta = 1$, which ensures that the solution is always feasible.

Now let $e = (u, v)$ be some edge. Assume, without loss of generality that x^v and x^u differ in the first two coordinates. Clearly, this edge can only be cut when we assign exactly one of u, v to t_1 or to t_2. This cannot happen with t_3 or t_4, because $x_3^v = x_3^u$ and $x_4^v = x_4^u$. Now we are only considering the probability that the edge is cut by assigning exactly one of u, v to t_1, and the second case is symmetrical. We assume that $x_1^v > x_1^u$.

Recall that we can assume that either both x_1^u and x_1^v are $\geq \Delta$, or they are both $\leq \Delta$.

If the first case is true, the edge can be cut by assigning v or u to t_1 only during the first two cuts the algorithm performs (if $\sigma_1 = 1$ or $\sigma_2 = 1$, respectively), but it cannot be cut during the third cut of the algorithm (because even if $\sigma_3 = 1$, we choose $\rho_3 \in (0, \Delta)$, and both x_1^v and x_1^u are $\geq \Delta$).

If the second case is true, the edge can be cut by assigning v or u to t_1 only during the third cut of the algorithm, if $\sigma_3 = 1$. This is because $\rho_1, \rho_2 > \Delta$, and $x_1^v, x_2^v \leq \Delta$.

Finally, we must set the distribution of ρ_3 (we only said that $\rho_3 \in (0, \Delta)$).

Suppose that $x_1^v, x_1^u \leq \Delta$. If $\sigma_3 = 1$, the probability that the edge will be cut by assigning v or u to t_1 during the third cut of the algorithm is:

$$\leq Pr\left(\begin{array}{c} \text{u was not assigned} \\ \text{on previous steps} \end{array} \middle| \; \sigma_3 = 1 \right) \cdot Pr(\rho_3 \in [x_1^u, x_1^v))$$

We will show later that the probability that u was not assigned on previous steps is $\leq \frac{l+2x_1^u}{3l}$. The density function of the distribution of ρ_3, which we denote by $P(x)$, is proportional to $\frac{1}{l+2x}$.

The Rounding Algorithm: We set two parameters: $\Delta = \frac{e^{\frac{1}{3}}-1}{3e^{\frac{1}{3}}-1}$, $l = 1 - 3\Delta = \frac{2}{3e^{\frac{1}{3}}-1}$.

We use an algorithm of the form Alg. We must specify the distributions of ρ_1, ρ_2, ρ_3.

We choose ρ_1 uniformly at random from $(\Delta, 1 - 2\Delta = l + \Delta)$, and set $\rho_2 = 1 - \Delta - \rho_1$. The parameter ρ_3 is chosen from $(0, \Delta)$ with density function $P(x) = \frac{2}{\ln \frac{l+2\Delta}{l}} \cdot \frac{1}{l+2x}$.

Note that $\rho_1 + \rho_2 + \rho_3 \leq 1 - \Delta + \Delta = 1$ and so the solution the algorithm returns is feasible.

Analysis: We show that for each edge e of length ϵ, the probability to be cut by the algorithm is $\frac{3e^{\frac{1}{3}}-1}{2} \cdot \epsilon$, and so the approximation ratio of the algorithm is $\frac{3e^{\frac{1}{3}}-1}{2} \approx 1.5934187$.

Let $e = (u, v)$ be some edge of length ϵ. We assume without loss of generality that the two vertices differ in the first two coordinates and that $x_1^v > x_1^u$. Thus $|x_1^u - x_1^v| = |x_2^u - x_2^v| = \epsilon$.

Clearly, this edge can be cut only during assignments to t_1 or t_2. We compute separately the probabilities that the edge is cut during assignments to t_1 and to t_2.

First, we compute the probability that the edge was cut during the assignment to t_1. We take care of 2 cases:

1. $x_1^u \geq \Delta$. We assume that $x_1^v \geq \Delta$, too. If $\sigma_1 = 1$ (which happens with probability $\frac{1}{4}$), the edge is cut with the probability $\frac{\epsilon}{l}$. The same analysis holds if $\sigma_2 = 1$. If $\sigma_3 = 1$, the edge cannot be cut, because $\rho_3 < \Delta$. So the probability that the edge is cut during the assignment to t_1 is $\frac{1}{4} \cdot 2 \cdot \frac{\epsilon}{l} = \frac{\epsilon}{2l} = \frac{3e^{\frac{1}{3}}-1}{4} \cdot \epsilon$.

2. $x_1^u \leq \Delta$. Again, we assume that $x_1^v \leq \Delta$, too. First, note that if $\sigma_1 = 1$ or $\sigma_2 = 1$, the edge cannot be cut during the assignment to t_1, since $\rho_1, \rho_2 > \Delta$.

   Now assume $\sigma_3 = 1$ (which happens with probability $\frac{1}{4}$). The probability that the edge will be cut during the assignment to t_1 in this case is less than or equal to

   $$Pr\left(\begin{array}{c} \text{u was not assigned} \\ \text{in previous steps} \end{array} \middle| \sigma_3 = 1 \right) \cdot \int_{x_1^u}^{x_1^v} P(x)dx \quad \leq$$

   $$Pr\left(\begin{array}{c} \text{u was not assigned} \\ \text{in previous steps} \end{array} \middle| \sigma_3 = 1 \right) \cdot P(x_1^u)\epsilon$$

   This is true because $P(x)$ is monotonous decreasing in x.

   We will show later that if $\sigma_3 = 1$, then u is assigned to some terminal after the first two cuts the algorithm performs with probability $\geq \frac{2l-2x_1^u}{3l}$. Thus the probability that u was not assigned in previous steps is $\leq 1 - \frac{2l-2x_1^u}{3l} = \frac{l+2x_1^u}{3l}$.

   So the probability that the edge is cut during the assignment to t_1 is $\leq$

$$Pr(\sigma_3 = 1) \cdot Pr\left(\begin{array}{c} u \text{ was not assigned} \\ \text{previously} \end{array} \middle| \sigma_3 = 1 \right) \cdot P(x_1^u)\epsilon \leq$$

$$\frac{1}{4}\frac{l+2x_1^u}{3l} \cdot \frac{2\epsilon}{(l+2x_1^u)\ln\frac{l+2\Delta}{l}} = \frac{\epsilon}{6l\ln\frac{l+2\Delta}{l}} = \frac{3e^{\frac{1}{3}}-1}{4} \cdot \epsilon$$

So the edge can be cut during assignment to t_1 with probability $\frac{3e^{\frac{1}{3}}-1}{4} \cdot \epsilon$. By the symmetry, the edge can be cut during assignment to t_2 with the same probability. So the probability the edge is cut by the algorithm is $\leq 2 \cdot \frac{3e^{\frac{1}{3}}-1}{4} \cdot \epsilon = \frac{3e^{\frac{1}{3}}-1}{2} \cdot \epsilon \approx 1.5934187\epsilon$.

Now it only remains to show that if $\sigma_3 = 1$ and $x_1^u \leq \Delta$, the probability that u is assigned after the first two cuts of the algorithm is $\geq \frac{2l-2x_1^u}{3l}$.

Claim 1 *Suppose $x_{\sigma_1}^u, x_{\sigma_2}^u < 1-2\Delta$. Then if $x_{\sigma_1}^u + x_{\sigma_2}^u \geq 1-\Delta$, u is assigned after the first two cuts of the algorithm with probability 1. If $x_{\sigma_1}^u + x_{\sigma_2}^u < 1-\Delta$, u is assigned after the first two cuts of the algorithm with probability $\frac{x_{\sigma_1}^u + x_{\sigma_2}^u - 2\Delta}{l}$.*

Proof:

1. Suppose $x_{\sigma_1}^u + x_{\sigma_2}^u \geq 1-\Delta$ and u was not assigned after the first two cuts of the algorithm, Then $x_{\sigma_1}^u < \rho_1$, $x_{\sigma_2}^u < \rho_2$. But $1-\Delta \leq x_{\sigma_1}^u + x_{\sigma_2}^u < \rho_1 + \rho_2 = 1-\Delta$. Contradiction.

2. Suppose $x_{\sigma_1}^u + x_{\sigma_2}^u < 1-\Delta$. Then the probability that u is assigned after the first two cuts of the algorithm is:

$$Pr[\rho_1 \leq x_{\sigma_1}^u] + Pr[\rho_2 \leq x_{\sigma_2}^u] - Pr[\rho_1 \leq x_{\sigma_1}^u \text{ and } \rho_2 \leq x_{\sigma_2}^u]$$

   Since ρ_1 is distributed uniformly in $(\Delta, 1-2\Delta)$, $x_1^u < 1-2\Delta$, and $1-3\Delta = l$, $Pr[\rho_1 \leq x_{\sigma_1}^u] \geq \frac{x_{\sigma_1}^u - \Delta}{l}$. Note that it is possible that $x_{\sigma_1}^u < \Delta$. In this case $Pr[\rho_1 \leq x_{\sigma_1}^u] = 0$, and $\frac{x_{\sigma_1}^u - \Delta}{l} < 0$.

   The same computation holds for $Pr[\rho_2 \leq x_{\sigma_2}^u] \geq \frac{x_{\sigma_2}^u - \Delta}{l}$.

   Finally, we compute $Pr[\rho_1 \leq x_{\sigma_1}^u \text{ and } \rho_2 \leq x_{\sigma_2}^u]$. Since $\rho_2 = 1-\Delta-\rho_1$, this is equivalent to $Pr[x_{\sigma_1}^u \geq \rho_1 \geq 1-\Delta-x_{\sigma_2}^u] = 0$, because $x_{\sigma_1}^u + x_{\sigma_2}^u < 1-\Delta$ and thus $x_{\sigma_1}^u < 1-\Delta-x_{\sigma_2}^u$.

   So, the probability that u is assigned after the first two cuts of the algorithm is $\geq \frac{x_{\sigma_1}^u - \Delta}{l} + \frac{x_{\sigma_2}^u - \Delta}{l} - 0 = \frac{x_{\sigma_1}^u + x_{\sigma_2}^u - 2\Delta}{l}$

□

Claim 2 *Suppose $\sigma_3 = 1$ and $x_1^u \leq \Delta$. Then the probability that u is assigned after the first two cuts of the algorithm is $\geq \frac{2l-2x_1^u}{3l}$.*

Proof:

First, if for any $i \in \{2,3,4\}$, $x_i^u \geq 1-2\Delta$, the probability that u is assigned to some terminal by the first two cuts is $\frac{2}{3}$. (This is the probability that i will be chosen as one of the first two terminals. If it happens, u is surely assigned because $\rho_1, \rho_2 < 1-2\Delta$.) Note that $\frac{2}{3} \geq \frac{2l-x_1^u}{3l}$. So now we assume that for each $i \in \{2,3,4\}$, $x_i^u < 1-2\Delta$. We take care of four different cases:

Case 1: For all three pairs of coordinates $(i,j) \ : \ i,j \in \{2,3,4\}$ have $x_i^u + x_j^u < 1-\Delta$. In this case, the probability that the vertex is assigned is $\frac{1}{3}(\frac{x_2^u+x_3^u-2\Delta}{l} + \frac{x_2^u+x_4^u-2\Delta}{l} + \frac{x_3^u+x_4^u-2\Delta}{l}) = \frac{2x_2^u+2x_3^u+2x_4^u-6\Delta}{3l} = \frac{2-2x_1^u-6\Delta}{3l} = \frac{2l-2x_1^u}{3l}$.

Case 2: There is exactly one pair of coordinates $(i,j) \ : \ i,j \in \{2,3,4\}$ such that $x_i^u + x_j^u \geq 1-\Delta$. We assume, without loss of generality, that the pair of the coordinates is $(2,3)$, and so $x_2^u + x_3^u \geq 1-\Delta$. Note that $x_4^u \leq \Delta$ must hold. If t_2 and t_3 are the first two coordinates, which happens with probability $\frac{1}{3}$, u is surely assigned. If $(2,4)$ is the first pair of coordinates, u is assigned with probability $\geq \frac{x_2^u-\Delta}{l}$, because $x_4^u < \Delta$. The same holds for the third pair of coordinates. So the probability u is assigned is:

$$\frac{1}{3}(1 + \frac{x_2^u-\Delta}{l} + \frac{x_3^u-\Delta}{l}) \geq \frac{l+x_2^u+x_3^u+x_4^u-3\Delta}{3l} = \frac{l+1-x_1^u-3\Delta}{3l} = \frac{2l-x_1^u}{3l} \geq \frac{2l-2x_1^u}{3l}$$

Case 3: There are exactly two pairs of coordinates $(i,j) \ : \ i,j \in \{2,3,4\}$ such that $x_i^u + x_j^u \geq 1-\Delta$. If any of these two pairs is chosen to be the first two coordinates (which happens with probability $\frac{2}{3}$), u will be assigned. So the probability that u is assigned is $\geq \frac{2}{3} \geq \frac{2l-2x_1^u}{3l}$.

Case 4: All the three pairs of coordinates have $x_i^u + x_j^u \geq 1-\Delta$. This case is impossible, because it means that $x_2^u + x_3^u + x_4^u \geq \frac{3}{2}(1-\Delta) = \frac{3e^{\frac{1}{3}}}{3e^{\frac{1}{3}}-1} > 1$.

□

Chapter 4

Upper Bounds for the Uniform Labeling Problem

The Linear Program

We again represent each graph node v by a vector $x^v \in \mathbb{R}^k$. The vector x^v has $x_i^v = 1$ if v is assigned to the terminal t_i. Otherwise, $x_i^v = 0$. We also introduce vector variables $z^{(u,v)} \in \mathbb{R}^k$ for each $(u,v) \in E$, with values $z^{(u,v)} = |x^u - x^v|$.

$$(UIP) \quad Min \ \tfrac{1}{2}\Sigma_{(u,v)\in E} w_{(u,v)} \cdot (\Sigma_{i=1}^k z_i^{(u,v)}) + \Sigma_{v\in V, t_i \in T} c(v,t_i) \cdot x_i^v$$

$$\begin{aligned}
s.t. \quad & x^{t_i} = e^i && i = 1,\ldots,k && (4.1)\\
& \Sigma_{i=1}^k x_i^v = 1 && \forall v \in V && (4.2)\\
& z_i^{(u,v)} \geq x_i^v - x_i^u && \forall (u,v) \in E,\ i = 1,\ldots,k && (4.3)\\
& z_i^{(u,v)} \geq x_i^u - x_i^v && \forall (u,v) \in E,\ i = 1,\ldots,k && (4.4)\\
& x_i^v \in \{0,1\} && \forall v \in V, i = 1,\ldots,k && (4.5)
\end{aligned}$$

Again, since the constraints (4.3), (4.4) ensure that $z_i^{(u,v)} \geq |x^u - x^v|$ and since we would like to decrease $z_i^{(u,v)}$ as much as we can, in the optimal solution $z_i^{(u,v)} = |x^u - x^v|$, and the objective function is actually

$$Min \ \frac{1}{2}\Sigma_{(u,v)\in E} w_{(u,v)} \cdot \|x^u - x^v\| + \Sigma_{v\in V, t_i \in T} c(v,t_i) \cdot x_i^v$$

Clearly, any feasible solution for (UIP) places all the graph nodes on the simplex vertices. For each $(u,v) \in E$, $\frac{1}{2}\|x^u - x^v\| = 0$ if u,v are assigned to the same simplex vertex, and $\frac{1}{2}\|x^u - x^v\| = 1$ otherwise. Thus, we pay $w_{u,v}$ for an edge (u,v) iff u and v are assigned to different terminals. For each node $v \in V$ and for each terminal $t_i \in T$, if v is assigned to t_i, $x_i^v = 1$, otherwise $x_i^v = 0$. So we pay the assignment cost $c(v,t_i)$ iff v is assigned to t_i. Thus, any feasible solution to (UIP) gives us a solution to the uniform labeling problem of the same or smaller cost. Given any feasible solution to uniform labeling problem, it is easy to construct a solution to (UIP) of the same cost by placing all the graph nodes on the simplex vertices corresponding to the terminals they are assigned to, and setting $z^{(u,v)} = |x^u - x^v|$. We denote by (ULP) the linear program obtained from (UIP) by replacing the integrality constraints (4.5) with

$$x_i^v \geq 0 \quad \forall v \in V, i = 1, \ldots k \tag{4.6}$$

Kleinberg and Tardos proved in [8] that the integrality gap of (UIP) is $\geq 2 - \frac{2}{k}$ for all k. We showed in the previous chapters that the integrality gap of the relaxation we used for the restricted multiway cut problem is $\geq 2 - \frac{2}{k}$. The same example we used for proving the integrality gap of the restricted multiway cut problem can be used here to prove the integrality gap for the uniform labeling problem. We only need to "translate" this example to the uniform labeling problem by specifying the vertex assignment costs. We set them in the following way: for each $v \in V \setminus T$, $t \in T$, $c(v,t) = 0$ if $t \in l_v$, and $c(v,t) = \Sigma_{e\in E} w_e + 1$ otherwise.

The next lemma shows that the approximation algorithms we used for the restricted multiway cut problem can be used to round the fractional solution of the uniform labeling problem, obtained from (ULP), achieving the same approximation ratios.

Lemma 4 *The two rounding algorithms for the restricted multiway cut problem for $k = 3$ and for $k = 4$ presented in the previous sections can be used*

for approximation of the uniform labeling problem for the same values of k, achieving the same approximation factors.

Proof: Let A be any of the two rounding algorithms from the previous sections, with approximation ratio α (while $\alpha = \frac{4}{3}$ for the algorithm for $k = 3$ and $\alpha = \frac{3e^{\frac{1}{3}}-1}{2}$ for the algorithm for $k = 4$). We use the fact that for any edge e of length ϵ, the probability that A will cut e is $\leq \epsilon\alpha$. (This fact was proved for both algorithms while analyzing their approximation ratios). Another fact we use is that the distribution of the cuts A chooses is independent of its input.

Suppose we are given an optimal fractional solution S to the uniform labeling problem obtained from solving (ULP). We denote by c_1 its vertex assignment cost, and by c_2 its edge stretch cost. The total cost of S is c_1+c_2. We use A to round this fractional solution and to obtain integer solution $A(S)$. The cost of the solution $A(S)$ also consists of 2 parts: vertex assignment cost which we denote by a_1 and edge stretch cost which we denote by a_2. We need to prove that $a_1 + a_2 \leq \alpha(c_1 + c_2)$. Since A is a factor-α rounding algorithm for restricted multiway cut problem, we know that $a_2 \leq \alpha c_2$. It remains to show that $a_1 \leq \alpha c_1$.

Claim 3 *For each $v \in V$ and for each $t_i \in T$, the probability that A assigns v to t_i is $\leq \alpha x_i^v$.*

Proof: Suppose, by contradiction, that the claim is wrong. Then there exist $v \in V$, $i \in \{1 \ldots k\}$, such that the probability that A assigns v to t_i is $> \alpha x_i^v$. We can add to the fractional solution a new vertex u on the facet spanned by $T \setminus \{t_i\}$, and a new edge (v, u), such that $\frac{1}{2}\|x^v - x^u\| = x_i^v$, and $w_{(v,u)} = 1$. Note that A cannot assign u to t_i. So if A assigns v to t_i, the edge (u, v) is surely cut. We know that the probability that A cuts the edge must be $\leq \alpha \cdot \frac{1}{2} \cdot \|x^u - x^v\| = \alpha x_i^v$, and also that this probability is greater than or equal to the probability of assigning v to t_i. Thus v is assigned to t_i with probability $\leq \alpha x_i^v$. Since the distribution of the cuts does not depend on the input, adding the edge could not change the probability of assigning v to t_i. This is a contradiction to the fact that before adding the edge, A assigned v to t_i with probability $> \alpha x_i^v$.

Note that we can add an edge (v, u) from v to the faced spanned by $T \setminus \{t_i\}$ of length x_i^v, by defining:

$x^u = (x_1^v + d_1, \cdots, x_{i-1}^v + d_{i-1}, 0, x_{i+1}^v + d_{i+1}, \cdots, x_k^v + d_k)$

while $\Sigma_{j \neq i} d_j = x_i^v$ and for each $j \neq i$, $x_j^v + d_j \leq 1$. This is possible because $\Sigma_{j=1}^k x_j^v = 1$. □

We now complete the proof of Lemma 4. The expected value of a_1 is the sum of expected values of the vertex assignment costs of all the vertices. For each $v \in V$, the expected value of the assignment cost is
$Pr[v \text{ is assigned to } t_i] \cdot c(v, t_i) \leq \Sigma_{t_i \in T} \alpha x_i^v c(v, t_i)$, and so the expected value of a_1 is $\leq \Sigma_{v \in V, t_i \in T} \alpha x_i^v c(v, t_i) = \alpha c_1$.
□

Bibliography

[1] G. Călinescu, H. Karloff, Y. Rabani. An Improved Approximation Algorithm for Multiway Cut. *Journal of Computer and System Sciences*, to appear. Preliminary version appeared in *Proc. STOC*, 1998, pp. 48–52.

[2] W.H. Cunningham and L. Tang. Optimal 3-terminal Cuts and Linear Programming. In *Proc. MPS Conference on Integer Programming and Combinatorial Optimization*, 1999.

[3] E.Dahlhaus, D.S. Johnson, C.H. Papadimitriou, P.D. Seymour, and M. Yannakakis. The Complexity of Multiway Cuts. *SIAM Journal of Computing*, 23(1994), pp. 864–894.

[4] L.R. Ford and D.R. Fulkerson. *Flows in Networks.* Princeton University Press, Princeton, NJ, 1962.

[5] A. Gupta and É. Tardos. A Constant Factor Approximation Algorithm for a Class of Classification Problems. In *Proc. STOC*, 2000, pp. 652–658.

[6] D.R. Karger, P. Klein, C. Stein, M. Thorup N.E. Young. Rounding Algorithms for a Geometric Embedding of Minimum Multiway Cut. In Proc. *STOC*, 1999, pp. 668-678.

[7] N. Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 1984, pp. 373-395.

[8] J.Kleinberg, É. Tardos. Approximation Algorithms for Classification Problems with Pairwise Relationships; Metric Labeling and Markov Random Fields. In *Proc. FOCS*, 1999, pp. 14–23.

[9] G.L.Nemhauser and L.A. Wolsey. *Integer and Combinatorial Optimization.* John Wiley & Sons, New York, NY, 1988.

אלגוריתמי קירוב לבעיות חתכים קשות

חיבור על מחקר

לשם מילוי חלקי של הדרישות לקבלת התואר
מגיסטר למדעים במדעי המחשב

יוליה צ'וז'וי

הוגש לסנט הטכניון — מכון טכנולוגי לישראל
אלול תש"ס חיפה ספטמבר 2000

המחקר נעשה בהנחיית דר' יובל רבני בפקולטה למדעי המחשב

ברצוני להודות ליובל רבני על הנחייתו המסורה במהלך המחקר

אני מודה לבעלי לאוניד על העידוד והתמיכה

אני מודה לטכניון על התמיכה הכספית הנדיבה בהשתלמותי

תוכן עניינים

רשימת איורים

תקציר מורחב

בבעיות של מתן תויות, נתון אוסף של עצמים ואוסף של תויות. נרצה לתת לכל עצם תוית כלשהי באופן עקבי עם ידע מוקדם שיש בידינו. נתמקד בשני סוגים של ידע מוקדם:

1. לכל עצם ולכל תוית, אנו יודעים להעריך, עד כמה סביר שהעצם צריך לקבל את התוית.
2. יש בידינו ידע מוקדם לגבי יחסים בין זוגות העצמים: אוסף של זוגות העצמים, כשלגבי כל זוג מהאוסף אנו סבורים כי שני העצמים שבזוג צריכים לקבל אותה תוית.

נרצה לתת לכל עצם תוית כלשהי, ובכל פעם שמתן התויות אינו עקבי עם הידע המוקדם, נשלם "קנס". נחפש דרך למתן תויות עם קנס מינימלי, ובצורה כזאת נקווה להגיע לפיתרון עקבי ככל האפשר עם הידע המוקדם. בעית מתן התויות במטריקה אחידה היא בעית מתן תויות עם ידע מוקדם שמתאים לשני הסוגים המתוארים לעיל.

בעית מתן התויות במטריקה אחידה מוגדרת באופן הבא: נתון גרף לא מכוון $G=(V,E)$, וקבוצה $T=\{t_1,\ldots t_k\}\subset V$ של צמתים מיוחדים שנקראים טרמינלים. בנוסף, לכל קשת $e\in E$ נתון משקל $w_e\geq 0$, ולכל צומת $v\in V$ וטרמינל $t\in T$ נתון מחיר השמה $c(v,t)$. פיתרון לבעיה הוא פונקצית השמה $f:V\backslash T\rightarrow T$. לכל פיתרון כזה מוגדר מחיר, ונחפש פיתרון בעל מחיר מינימלי. מחיר הפיתרון מורכב משני חלקים:

1. מחיר של השמת הצמתים: $\Sigma_{v\in V}c(v,f(v))$ (כלומר, אם צומת v מקבל השמה t, הוא צריך לשלם $c(v,t)$).
2. מחיר של חיתוך הקשתות: כל קשת $(u,v)\in E$, ששני קצוותיה מקבלים השמות שונות, צריכה לשלם $w_{(u,v)}$. סך הכל מחיר חיתוך הקשתות הוא: $\Sigma_{\substack{(u,v)\in E\\ f(u)\neq f(v)}} w_{(u,v)}$.

המחיר הכולל של פתרון הוא הסכום של מחיר השמת הצמתים ומחיר חיתוך הקשתות.

כל בעיה של מתן תויות עם ידע מוקדם, שמתאים לשני הסוגים המתוארים לעיל, ניתנת לתרגום לבעיה זו באופן הבא: העצמים יתורגמו לצמתים הרגילים והתויות

לטרמינלים. ניתן לראות שפונקציית ההשמה מתאימה לפונקציית מתן התויות לעצמים. הידע המוקדם מהסוג הראשון יתורגם למחירי השמות של הצמתים: לכל צומת v וטרמינל t, ככל שאנו סבורים יותר שהעצם המתאים ל-v צריך לקבל את התוית המתאימה ל-t, כך מחיר ההשמה $c(v,t)$ יהיה נמוך יותר (הקנס שנשלם על ההשמה יהיה נמוך יותר). הידע המוקדם מהסוג השני יתורגם למחירי חיתוך הקשתות: לכל זוג צמתים (u,v) שיש לנו ידע מוקדם מהסוג השני לגבי זוג העצמים המתאים, תהיה קשת בגרף בין זוג הצמתים. ככל שאנו סבורים יותר שזוג העצמים המתאים צריך לקבל תויות זהות, כך משקל הקשת יהיה גבוה יותר (ונצטרך לשלם קנס גבוה יותר אם שני העצמים מקבלים תויות שונות).

ניתן לראות כי בעית מתן התויות במטריקה אחידה שהגדרנו לעיל היא בעיה מאוד כללית, וישנן הרבה בעיות בתחומים שונים של מדעי המחשב שניתן לתרגם אותן לבעיה זו.

עבור המקרה הפרטי בו $k = 2$ (ישנם רק שני טרמינלים), ניתן לתרגם את בעית מתן התויות במטריקה אחידה לבעית חתך בין שני צמתים, אשר ניתנת לפיתרון יעיל. אולם לכל $k \geq 3$, בעית מתן התויות במטריקה אחידה היא NP-קשה, וזה אומר כי קרוב לוודאי שאין אלגוריתם יעיל לפתרון הבעיה. דרך אחת להתמודד עם בעיות כאלה היא מציאת אלגוריתם קירוב עבורן. אלגוריתם קירוב בעל פקטור קירוב α הוא אלגוריתם יעיל שמובטח לגבינו כי מחיר הפיתרון שהוא מחזיר הוא לכל היותר פי α גדול יותר ממחיר הפיתרון האופטימלי לבעיה.

תוצאת קושי נוספת לבעיה זו קובעת כי הבעיה היא Max SNP-קשה לכל $k \geq 3$. זה אומר כי קרוב לוודאי שלא קיימת סכימת קירוב פולינומית עבור בעיה זו. במילים אחרות, לא יתכן שלכל ϵ קיים אלגוריתם קירוב בעל פקטור $1+\epsilon$. לכן הדבר הטוב ביותר שאנו יכולים לקוות לו הוא אלגוריתם קירוב בעל פקטור קבוע. תוצאת הקירוב הטובה ביותר שהיתה ידועה לגבי בעיה זו היא פקטור קירוב 2. בעבודה זו התמקדנו בשני מקרים פרטיים של הבעיה. המקרה הראשון הוא כאשר $k = 3$ (k הוא מספר הטרמינלים). במקרה זה הצלחנו להשיג פקטור קירוב של $\frac{4}{3}$. המקרה השני הוא כאשר $k = 4$, ובמקרה זה פקטור הקירוב אותו השגנו הוא: $\frac{3e^{\frac{1}{3}}-1}{2} \approx 1.5935$.

בעיה חשובה שהיא קרובה לבעיה שהגדרנו היא בעית החתך הרב צדדי. בבעיה זו נתון גרף לא מכוון $G = (V,E)$ עם אוסף של צמתים מיוחדים $T = \{t_1, \ldots, t_k\} \subset V$ שנקראים טרמינלים, ומשקל אי שלילי w_e לכל קשת $e \in E$. המטרה היא שוב מציאת פונקצית השמה $f : V \backslash T \to T$ בעלת מחיר מינימלי, כאשר המחיר של פונקציית השמה הוא סכום משקלי הקשתות שקצוותיהן הושמו לטרמינלים שונים. ניתן לראות בעיה זו כמקרה פרטי של בעית מתן התויות במטריקה אחידה, כאשר כל המחירים של השמות הצמתים הם $c(v,t) = 0$. גם בעיה זו ניתנת לפיתרון יעיל

3. השלב האחרון הוא עיגול הפיתרון של התוכנית הליניארית, לשם קבלת פיתרון של התוכנית בשלמים, שהוא גם פיתרון לבעית מתן התויות. המטרה של שלב זה היא עיגול הפיתרון של התוכנית הליניארית עם הפסד קטן ככל האפשר במחיר הפיתרון. הפסד זה במחיר הפיתרון יקבע את פקטור הקירוב.

בעבודה זו השתמשנו בתוכנית בשלמים ובתוכנית הליניארית בהן השתמשו לקביעת התוצאה הקודמת לבעית מתן התויות במטריקה אחידה. עיקר העבודה הוא מציאת אלגוריתמים בעלי פקטורי קירוב טובים יותר לעיגול פיתרונות של התוכנית הליניארית למקרים המיוחדים של $k = 3$ ו-$k = 4$.