Approximation Algorithms for Clustering

Anthony Ian Wirth

A dissertation

presented to the faculty

of Princeton University

in Candidacy for the Degree

of Doctor of Philosophy

Recommended for Acceptance

by the Department of

Computer Science

January 2005

© Copyright by Anthony Ian Wirth, 2004. All rights reserved.

Abstract

Approximation Algorithms for Clustering

Clustering items into groups is a fundamental problem in the information sciences. Many typical clustering optimization problems are NP-hard and so cannot be expected to be solved optimally in a reasonable amount of time. Although the use of heuristics is common, in this dissertation we seek approximation algorithms, whose performance ratio in relation to the optimal solution can be guaranteed and whose running time is a polynomial function of the problem instance size.

We start by examining variants of the asymmetric k-center problem. We demonstrate an $O(\log^* n)$-approximation algorithm for the asymmetric *weighted* k-center problem. Here, the vertices have weights and we are given a total budget for opening centers. In the *p-neighbor* variant each vertex must have p (unweighted) centers nearby: we give an $O(\log^* k)$-bicriteria algorithm using $2k$ centers, for small p. We also show that the following three versions of the asymmetric k-center problem are inapproximable: *priority k-center*, *k-supplier*, and *outliers with forbidden centers*.

The bulk of the dissertation concerns *correlation clustering*: clustering a collection of elements based on pairwise judgments of similarity and dissimilarity. The problem in-

stance does not include a distance relation between the elements. We partition the elements into clusters so that the number of pairs correctly (resp. incorrectly) classified with respect to the input judgment labeling is maximized (resp. minimized). It is worthwhile studying both complete instances, in which every pair is labeled, and general instances, in which some input pairs might not have labels.

Specifically, we demonstrate a factor 4 approximation for minimization on complete instances, and a factor $O(\log n)$ approximation for general instances. For the maximization version, we give a factor 0.7664 approximation for general instances, noting that a PTAS is unlikely by proving APX-hardness. We also prove the APX-hardness of minimization on complete instances.

We provide the first nontrivial approximation algorithm for maximizing the correlation: the difference between the number of pairs correctly classified and the number incorrectly classified. The factor $\Omega(1/\log n)$ algorithm is derived from an approximation algorithm for maximizing a fairly general type of quadratic program on the unit hypercube.

Acknowledgements

First and foremost I would like to thank Moses Charikar for his support, patience, advice, friendship, and his brilliant mind. It was a pleasure being his research student and a stimulating and exciting experience being part of his research group.

I am most grateful to Sanjeev Arora for recruiting me to Princeton, acting as my advisor during my first year, and introducing me to approximation algorithms.

I greatly appreciate the two opportunities that Bob Tarjan gave me to work for and learn from him, as a teaching assistant and as a lab intern at Hewlett-Packard.

The quality and fun of my PhD research was considerably enhanced by the many interactions with my excellent coauthors: Amit Chakrabarti†, Inge Li Gørtz, Venkatesan Guruswami, Yaoyun Shi†, and Andrew Yao†.

I am especially indebted to Elad Hazan and Satyen Kale for sharing their expertise with efficient semidefinite programming algorithms.

Amongst others, my research has been aided by communications and conversations with Noga Alon, Shuchi Chawla, Michael Dinitz, Ben Green, George Karakostas, Subhash Khot, Konstantin Makarychev, Yury Makarychev, Assaf Naor, and Renato Werneck.

I thank the reviewers of my papers for their helpful suggestions and insightful im-

provements.

I met many inspiring and fascinating people and made many friends at Princeton, but amongst these I would like to single out Amit Chakrabarti for his encouragement, mentoring, wonderful conversations, and his taste in music, film, and food.

To my best friend Chaitanya: I am glad that we have both had the American PhD experience and I am looking forward to your coming back home to Melbourne too.

And to my parents, Eve and Andrew, my sister Karen, and my grandmothers Susan and Stefa: I missed you deeply while I was away, but we are all together again now.

AIW

October 2004

I gratefully acknowledge NSF ITR grant CCR-0205594 (awarded to Moses Charikar), a DIMACS summer research fellowship (2002), a Dean's fund for scholarly travel award, and a Gordon Wu graduate fellowship for supporting my PhD research.

† The research related to our joint paper *Informational complexity and the direct sum problem for simultaneous message complexity* [Proc. of 42nd FOCS, pp270–8, 2001] does not appear in this dissertation.

This dissertation is dedicated to the memory of my grandfather Arthur Lindner, whom I wish I could have met.

Contents

Abstract **iii**

Acknowledgements **v**

1 Setting the scene **1**

1.1 The clustering problems in the dissertation 2

1.2 Approximation algorithms 4

1.3 Results highlights and organization 5

1.4 Notation and conventions 8

1.5 Prior publication 9

2 Asymmetry in k-center variants **10**

2.1 Introduction 10

2.2 Definitions 15

2.3 Asymmetric k-center review 18

2.4 Asymmetric weighted k-center 19

2.5 Asymmetric p-neighbor k-center 26

2.6 Inapproximability results 31

2.7 Pattern rotation 34

3 Correlation clustering **38**

3.1 Introduction 38

3.2 Previous and related work 40

3.3 Our results 41

4 Minimizing disagreements **46**

4.1 General graphs 46

4.2 Complete graphs 49

4.3 Approximation limitations 55

4.4 The connection to feedback edge sets 59

5 Maximizing agreements **65**

5.1 A linear program with poor integrality gap 65

5.2 Rounding a semidefinite program 66

5.3 Almost satisfiable instances 69

6 Maximizing correlation **72**

6.1 Reduction to the two cluster case 72

6.2 Maximizing quadratic programs 76

6.3 Approximating MAXCORR: the two cluster case 79

6.4 The relationship with max cut 84

6.5 Correlated random variables and distributions on cuts 86

7 Hardness of approximation **88**

7.1 MINDISAGREE in general graphs 88

7.2 MAXAGREE in general graphs 90

7.3 MINDISAGREE in complete graphs 94

8 Efficient approximation algorithms **101**

8.1 The MAXCORR SDP 101

8.2 MINDISAGREE in complete instances 106

9 What's next **109**

9.1 Improving the algorithms 109

9.2 Consensus clustering 111

9.3 Epilogue 112

List of Figures

2.1 A cover with half the weight: near and far centers. 22

2.2 Recursive greedy set cover 25

2.3 Reduction to asymmetric priority k-center 33

2.4 Simple pattern rotation instance 35

4.1 The two cases of ALGCOMPLETE 50

4.2 Charging mistakes to the LP costs in ALGCOMPLETE 53

4.3 MINDISAGREE instance with integrality gap almost 2 56

4.4 Region growing with fixed thresholds 58

4.5 Construction of a new NEPPC 62

6.1 Optimum versus singleton and two cluster solutions 73

7.1 Reduction from MAX 3SAT to MAXAGREE 92

7.2 Flower gadget for APX-hardness of complete MINDISAGREE 96

List of Tables

2.1 Overview of approximation results for k-center variants 14

3.1 Correlation clustering approximation results 42

Chapter 1

Setting the scene

Humans have a strong temptation to place objects, people and ideas into groups. It is hard to resist the apparent efficiency of stereotyping. For many scientific and computational problems, clustering is a key tool in ensuring data consistency, cost minimization and computational efficiency.

There is a myriad of problems in location theory that one might want to consider, such as where to open warehouses and where to place network hubs. These can often be modeled using the k-center, k-median and uncapacitated facility location frameworks.

Contemporary clustering problems include classifying documents and books, determining which images represent the same object and associating gene expression patterns with specific medical conditions in microarray experiments.

Many clustering problems, and their proposed solution methods, share certain desirable properties.

First, the databases or data sets under consideration are often very large. For this rea-

son it is imperative to have methods whose running times are slowly growing functions. All of the algorithms in this dissertation run in polynomial time in the input size. However, for enormous databases, quadratic or even linear time algorithms may be more appropriate. Some gain might be possible if it is known that certain parts of the input will never be clustered with others and so we can process each block at a time. If not, we might also have to consider algorithms that do not store much information, but instead process the data in a small number of passes.

We would like some confidence in the partitioning that emerges from our procedure. It should be relatively insensitive to small changes in the input data and to different (sensible) clustering schemes.

One of the purposes of clustering is to make other computational processes efficient. An object recognition tool might have a library with millions of images, some of which might be different views of the same object. It makes sense to preprocess the library into a number of clusters so that a new image need only be compared with a single representative image from each cluster.

1.1 The clustering problems in the dissertation

This dissertation presents a number of new approximation algorithms for some contemporary clustering problems. We open with asymmetric variants of the k-center problem. Being one of the standard location theory problems, we mention only that there has been a great deal of activity recently related to asymmetric versions.

The remainder of the dissertation relates to a fairly new clustering paradigm: *corre-*

lation clustering. Two key features of this framework are that the number of clusters to be built is not specified in the input and that instead of a distance function on the data items we are given advice as to which pairs of items are similar. Chapter 3 contains the necessary theoretical background; at this point we consider the position of correlation clustering in the world of clustering techniques and applications.

Uncapacitated facility location is the first example that comes to mind of a problem in which the number of clusters is an outcome of the solution procedure. The various hierarchical clustering procedures also share this property, though they suffer from a level of subjectivity as the user can choose where to *cut* the dendrogram. On the other hand they do provide a large amount of information in a single representation. We know how to produce a hierarchical clustering whose performance is only a constant factor worse than the optimum if the number of clusters had been specified in advance [24]. Another generic approach is to maximize the between-cluster variance while minimizing the within-cluster variance. This criterion alone would force every item into a single cluster, so it is essential to include a penalty for *model complexity* [63].

There are a number of clustering algorithms that take advice about which items are similar, possibly in addition to a distance relation between the items. This advice might take the form of hard constraints, that a pair of items must (or cannot) be clustered together, or perhaps soft constraints, that there is evidence that they should (not) be in one cluster [76]. It is generally assumed that this advice is self-consistent. In correlation clustering, however, we make no such assumption: this is one of the principal difficulties in the clustering task. If the data items come with a metric, another approach is to distort the metric so that similar pairs of items end up near each other [78]. Both Wagstaff *et*

al. [76] and Xing *et al.* [78] use the k-means procedure to derive the clustering. Although k-means is a fundamental technique, it is a heuristic and makes no guarantee about the performance of the algorithm in relation to the optimum.

The principal application for these advice-taking clustering processes is in data cleaning [12]. We might have a family of bibliographic records or different collections of medical records and we want to remove the duplicates, merge the relevant information and ensure consistency in the information [22]. Alternatively, you might have one list of people who are registered to vote and another list of people who are, for some reason, ineligible to vote. Your task is to remove the latter from the former, but how do you ensure that records refer to the same people? A further application is the coreference problem: an automated text analyzer must match pronouns with the nouns they refer to [60].

1.2 Approximation algorithms

So much has been said about approximation algorithms in the last fifteen years that it is hard to offer any new generic wisdom here. We therefore merely present a definition of the type of problem under examination and the notion of the approximation algorithm. There are a number of excellent compendiums [45, 77, 74] of approximation algorithms. Some early glimpses of how approximation algorithms might cope with NP-completeness are in Garey and Johnson's classic text [33].

Approximation algorithms are one of the strategies for coping with the apparent difficulty of many common optimization problems. A decision problem (one that requires a *yes* or *no* answer) is in the class NP if it is possible to verify that a *yes* instance is indeed

a *yes* instance in polynomial time. A decision problem is NP-hard if it as least as hard as any problem in the class NP: an instance of any problem in NP can be reduced to a corresponding instance of the NP-hard problem in polynomial time.

Alternatively, we can consider NP-optimization problems, in which each instance has a family of feasible solutions, each of which has an objective value. The feasibility and the objective value of a solution can be determined in polynomial time. This objective value is often some size, cost, length, or weight. For a minimization (or maximization) problem, an optimal solution is one with the smallest (or largest) objective value. We will call an NP-optimization problem NP-hard if the related decision problem (*Is there a feasible solution with objective value less/more than such-and-such?*) is NP-hard.

Finally, given any instance of a particular minimization problem, a factor α approximation algorithm for that minimization problem returns a feasible solution whose objective value is at most a factor α greater than the optimum, in polynomial time. Note that the algorithm guarantees that the result returned is within the specified factor of the optimum. We adopt the convention here of expressing the performance ratio of an approximation algorithm for a maximization problem with a quantity less than 1. That is to say, an approximation algorithm for a maximization problem returns a solution whose objective value is at least α multiplied by the optimum.

1.3 Results highlights and organization

This section provides concise definitions of the problems examined in this dissertation and a summary of the main results.

We start with the k-center problem. Given an n-vertex graph whose edges have costs, we are to find a subset of k vertices, called centers, so that the maximum distance of any vertex from its nearest center is minimized. In the asymmetric version we need not assume that the distance from x to y is that same as that from y to x. Alternatively, each vertex might have a weight assigned so that, rather than only being allowed k centers, there is a restriction on the total weight of centers.

Chapter 2

- An $O(\log^* n)$ approximation algorithm for the weighted asymmetric k-center problem. Previously, an $O(\log^* k)$ approximation was known for the unweighted problem [4], as well as an $\Omega(\log^* n)$ hardness result [21, 42, 20].

- An $O(\log^* k)$ approximation for the asymmetric p-neighbor k-center problem, for $p \leq n/k$.

The remainder of the dissertation is concerned with the correlation clustering framework, introduced in Chapter 3. There are three key problems, all closely related. Given a graph whose edges are labeled either $+$ or $-$, the aim is to find a clustering of the vertices so that the number of edges in disagreement with the clustering is minimized. The disagreements are the $+$ edges between clusters and the $-$ edges within clusters. Alternatively we might wish to maximize the number of agreements, or to maximize the correlation, which is the difference between the number of agreements and disagreements. We also consider general instances in which the edges have weights in addition to sign labels. This allows for the possibility of some edges having zero weight, effectively

implying that they are absent.

Chapter 4

- An $O(\log n)$ approximation for minimizing disagreements in general instances. Independently, but simultaneously, two other research groups proved this result [28, 25].
- A factor 4 approximation algorithm for minimizing disagreements in complete instances, improving on the previous factor 17433 algorithm [7].

Chapter 5

- A 0.7664 semidefinite program-based approximation for maximizing agreements in general instances, improving on the naive $1/2$ approximation.
- A method to obtain a solution with $1 - O(\sqrt{\varepsilon}\log(1/\varepsilon))$ proportion of the edges as agreements even if the optimal clustering has $1 - \varepsilon$ fraction of edges in agreement.

Chapter 6

An $\Omega(1/\log n)$ approximation for maximizing the correlation in general instances, a problem for which there were no previous significant results.

Chapter 7

- A reduction from minimum multicut to the problem of minimizing disagreements in complete instances, a fact also demonstrated by other groups [28, 25]. This sug-

gests that an $o(\log n)$ approximation algorithm is unlikely. Previously, this problem was known to be APX-hard [8].

- Related hardness of approximation results for maximizing agreements ($79/80 + \varepsilon$), minimizing disagreements ($29/28 - \varepsilon$) and maximizing correlation ($25/26 + \varepsilon$) in general instances.

- A proof that minimizing disagreements in complete instances is APX-hard.

Chapter 8

- An $\tilde{O}(mn)$ time procedure for solving the semidefinite program used to maximize the correlation, based on existing efficient algorithms for the max cut SDP [55]. Standard interior point techniques exhibit worst case $\tilde{O}(n^{3.5})$ time complexity.

- An $\tilde{O}(n^5)$ time procedure for solving the linear program used to minimize the disagreements in a complete instance. Standard interior point techniques require $O(n^8)$ time in the worst case.

1.4 Notation and conventions

Throughout this dissertation, n will stand for the number of items to be clustered, and m to the number of pairs of items that have some relevant property or constraint. When the problem can be represented as a graph, n is the number of vertices and m the number of edges.

To avoid any uncertainty, we note that $\log$ stands for $\log_2$ by default, while $\ln$ stands for $\log_e$.

For every integer $i > 1$, $\log^i x = \log(\log^{i-1} x)$, and $\log^1 x = \log x$. We let $\log^* x$ represent the smallest integer i such that $\log^i x \leq 2$.

Terms such as $\tilde{O}(\cdot)$ and $\Omega(\cdot)$ that describe the asymptotic behavior of functions are described in Chapter 3 of Cormen *et al.*'s text [23].

In correlation clustering diagrams, solid lines indicate *positive* edges, whereas dashed lines indicate *negative* edges.

1.5 Prior publication

Many of the results in this dissertation have appeared in prior publications [38, 14, 16]. The first two publications have, subject to minor amendment, been accepted to appear in *Theoretical Computer Science* and the *Journal of Computer and System Sciences*, respectively.

Chapter 2

Asymmetry in k-center variants

2.1 Introduction

Imagine that you have been given funding to establish a new terrorism-response team in a large city. You have a model of the time it takes to reach one point in the city from another. As part of a pledge to the public, you have guaranteed that the response time to any emergency will be at most 10 minutes. Although you have the power to seize land to establish the bases for the terror-response team, your budget only allows you to open k such bases. Can you meet this challenge? Where should you place the k bases to minimize the worst case response time?

This is the *k-center* problem—a type of clustering problem that is similar to the facility location [59] and k-median [6] problems. The motivation for the *asymmetric* k-center problem, in our example, is that traffic patterns or one-way streets might cause the travel time from one point to another to differ depending on the direction of travel. Although the k-center problem has traditionally been analyzed in the context of a metric, here we

retain the triangle inequality, but abandon the symmetry.

Symmetry is a vital concept in graph approximation algorithms. Recently, the k-center problem was shown to be $\Omega(\log^* n)$ hard to approximate [21, 42, 20], even though the symmetric version has a factor 2 approximation. Facility location and k-median both have constant factor algorithms in the symmetric case, but are provably $\Omega(\log n)$ hard to approximate without symmetry [3]. The traveling salesman problem is a little better, in that no super-constant hardness is known, but without symmetry no approximation algorithm better than $O(\log n)$ has been found either.

Definition 2.1 (k-center) *Given $G = (V, E)$, a complete graph with nonnegative (but possibly infinite) edge costs, and a positive integer k, find a set S of k vertices, called* centers, *with minimum covering radius. The covering radius of a set S is the minimum distance R such that every vertex in V is within distance R of some vertex in S.*

Note that the approximation ratio reflects the length of the radius, not the size of the cover, which is a fixed value k.

Kariv and Hakimi [51] showed that the k-center problem is NP-hard. Without the triangle inequality the problem is NP-hard to approximate within any factor (there is a straightforward reduction from the dominating set problem). We henceforth assume that the edge costs satisfy the triangle inequality. Hsu and Nemhauser [48], using the same reduction, showed that the metric k-center problem cannot be approximated within a factor of $(2 - \varepsilon)$ unless P = NP. In 1985 Hochbaum and Shmoys [46] provided a factor 2 algorithm, essentially the best possible, for the metric k-center problem. In 1996 Panigrahy and Vishwanathan [75, 66] gave the first approximation algorithm for the asymmetric

problem, with factor $O(\log^* n)$. Archer [4] proposed two $O(\log^* k)$ algorithms based on many of the ideas of Panigrahy and Vishwanathan. The complementary $\Omega(\log^* n)$ hardness result [21, 42, 20] shows that these approximation algorithms are asymptotically optimal.

Variants of the k-center problem

A number of variants of the k-center problem have already been explored in the context of symmetric graphs. An obvious first consideration is that some delivery hubs are more expensive to establish than others. Instead of a restriction on the number of centers we can use, in the *weighted k-center* problem each vertex has a weight and we are given a budget W, which limits the total weight of centers that are established. Hochbaum and Shmoys [47] produced a factor 3 algorithm for this problem, which has recently been shown to be tight [21, 20].

Hochbaum and Shmoys [47] also studied the *k-supplier* problem, where the vertex set is segregated into suppliers and customers. Only supplier vertices can be centers and only the customer vertices need to be covered. Hochbaum and Shmoys gave a 3-approximation algorithm and showed that it is the best possible.

Khuller *et al.* [54] investigated the *p-neighbor k-center* problem, where the aim is to minimize the distance of the p^{th} furtherest center. This problem is motivated by the need to account for facility failures: even if up to $p-1$ facilities fail, every demand point has a functioning facility nearby. They gave a 3-approximation algorithm for all p, and a best possible 2-approximation algorithm when $p < 4$, noting that the case where p is small is "perhaps the practically interesting case".

Maybe some demand points are more important than others. Plesnik [68] studied the *priority k-center* problem, in which the effective distance to a demand point is enlarged in proportion to its specified priority. Plesnik approximates the symmetric version within a factor of 2.

Charikar *et al.* [15] note that a disadvantage of the standard k-center formulation is that a few distant clients, *outliers*, can force centers to be located in isolated places. They suggest a variant of the problem, the k-center problem with *outliers and forbidden centers*, where a small subset of clients may be denied service, and some points are forbidden from being centers. Charikar *et al.* gave a (best possible) 3-approximation algorithm for the symmetric version of this problem.

Finally, Bhatia *et al.* [11] considered a network model, such as a city street network, in which the traversal times change as the day progresses. This is known as the k-center problem *with dynamic distances*: we wish to assign the centers so that the maximum distance of any vertex from its nearest center at any time of the day is minimized.

Results and organization

Table 2.1 gives an overview of the best known results for the various k-center problems. In this chapter, we explore some of the asymmetric variants that are not yet in the literature.

Section 2.2 contains the definitions and notation required to develop the results. In Section 2.3 we briefly review the algorithms of Panigrahy and Vishwanathan [66], and Archer [4]. The techniques used in the standard k-center problem are often applicable to the variants.

Table 2.1: An overview of the approximation results for k-center variants. The results presented in this chapter are in boldface. †The maximum ratio of an edge's greatest length to its smallest length is denoted by β. ‡This is a bicriteria algorithm using $k(1 + 3/(\nu+1))$ centers. §For $p < 4$. ¶This is a bicriteria algorithm using $2k$ centers, for $p \leq n/k$.

Problem	**Symmetric**		**Asymmetric**	
k-center	2	[46]	$O(\log^* k)$	[4]
k-center with dynamic distances	$1 + \beta$ †	[11]	$O(\log^* n + \nu)$ ‡	[11]
weighted k-center	3	[47]	$\mathbf{O(\log^* n)}$	[38]
p-neighbor k-center	3 (2 §)	[17]	$\mathbf{O(\log^* k)}$ ¶	[38]
priority k-center	2	[68]	**Inapproximable**	[38]
k-center with outliers and forbidden centers	3	[15]	**Inapproximable**	[38]
k-suppliers	3	[47]	**Inapproximable**	[38]

Our first result, in Section 2.4, is an $O(\log^* n)$ approximation for the asymmetric weighted k-center problem. In Section 2.5 we develop an $O(\log^* k)$ approximation for the asymmetric p-neighbor k-center problem, for $p \leq n/k$. As noted by Khuller *et al.* [54], the case where p is small is the most interesting case in practice. This a bicriteria algorithm, allowing $2k$ centers to be used rather than just k. Turning to hardness, we show that the asymmetric versions of the k-center problem with outliers (and forbidden centers), the priority k-center problem, and the k-supplier problem are NP-hard to approximate within any factor (Section 2.6).

Finally, in Section 2.7, we introduce *pattern rotation*, a combinatorial covering problem that we considered when attempting to prove a super-constant integrality gap for asymmetric k-center. We demonstrate an $\Omega(\log n)$ integrality gap for the pattern rotation cover size.

2.2 Definitions

The input to the asymmetric k-center problem is a distance function d on every *ordered* pair of vertices—distances are allowed to be infinite—and a bound k on the number of centers. Note that we assume that the edges are *directed*.

Definition 2.2 *Vertex c* covers *vertex v* within r, *or c r-covers v, if $d_{cv} \leq r$. We extend this definition so that a set C r-covers a set A if for every $a \in A$ there is some $c \in C$ such that c covers a within r. Often we abbreviate "1-covers" to "covers".*

Many of the algorithms for k-center and its variants do not, in fact, operate on graphs with edge costs. Rather, they operate on bottleneck graphs [47], in which every edge has

unit cost, but an edge is included only if its distance in the original graph is lower than some threshold. The only sensible thresholds are the $n(n-1)$ inter-vertex distance values. Consequently, many algorithms essentially run through a sequence of bottleneck graphs with these thresholds in ascending order. This can be thought of as *guessing* the optimal radius R_{OPT}. The approach works because the algorithm either returns a solution, within the specified factor of the current threshold radius, or it fails, in which case R_{OPT} must be greater than the current radius.

Definition 2.3 (Bottleneck graph G_r) *For $r > 0$, define the bottleneck graph G_r of the graph $G = (V, E)$ to be the graph $G_r = (V, E_r)$, where $E_r = \{(i, j) : d_{ij} \leq r\}$ and all edges have unit cost.*

Most of the following definitions apply to *bottleneck* graphs.

Definition 2.4 (Power of graphs) *The t^{th} power of a graph $G = (V, E)$ is the graph $G^{(t)} = (V, E^{(t)})$, $t > 1$, where $E^{(t)}$ is the set of ordered pairs of distinct vertices that have a path of at most t edges between them in G.*

Definition 2.5 *For $i \in \mathbb{N}$ define*

$$\Gamma_i^+(v) = \{u \in V \mid (v, u) \in E^{(i)}\} \cup \{v\}, \qquad \Gamma_i^-(v) = \{u \in V \mid (u, v) \in E^{(i)}\} \cup \{v\} \ ,$$

that is, in the bottleneck graph there is a path of length at most i from v to u, respectively u to v.

Notice that in a symmetric graph $\Gamma_i^+(v) = \Gamma_i^-(v)$. We extend this notation to sets so that $\Gamma_i^+(S) = \{u \in V \mid u \in \Gamma_i^+(v) \text{ for some } v \in S\}$, with $\Gamma_i^-(S)$ defined similarly. We use $\Gamma^+(v)$ and $\Gamma^-(v)$ instead of $\Gamma_1^+(v)$ and $\Gamma_1^-(v)$.

Definition 2.6 *For $i \in \mathbb{N}$ define*

$$\Upsilon_i^+(v) = \Gamma_i^+(v) \setminus \Gamma_{i-1}^+(v), \qquad \Upsilon_i^-(v) = \Gamma_i^-(v) \setminus \Gamma_{i-1}^-(v) \ ,$$

that is, the nodes for which the path distance from v is exactly i, and the nodes for which the path distance to v is exactly i, respectively.

For a set S, the extension follows the pattern $\Upsilon_i^+(S) = \Gamma_i^+(S) \setminus \Gamma_{i-1}^+(S)$. We use $\Upsilon^+(v)$ and $\Upsilon^-(v)$ instead of $\Upsilon_1^+(v)$ and $\Upsilon_1^-(v)$.

Definition 2.7 (Center capturing vertex [CCV]) *A vertex v is a* center capturing vertex *(CCV) if $\Gamma^-(v) \subseteq \Gamma^+(v)$, that is, v covers every vertex that covers v.*

In the graph $G_{R_{\mathsf{OPT}}}$ the optimum center that covers v must lie in $\Gamma^-(v)$; for a CCV v, it lies in $\Gamma^+(v)$, hence the name. In symmetric graphs all vertices are CCVs: this property leads to the standard 2-approximation [46].

The following three fundamental problems, related to k-center, are NP-complete [33].

Definition 2.8 (Dominating set) *Given a graph $G = (V, E)$, and a weight function $w : V \to \mathbb{Q}^+$ on the vertices, find a minimum weight subset $D \subseteq V$ such that every vertex $v \in V$ is covered by D, that is, $\Gamma^+(D) = V$.*

Definition 2.9 (Set cover) *Given a universe $\mathcal{U}$ of n elements, a collection $\mathcal{S} = \{S_1, \ldots, S_k\}$ of subsets of $\mathcal{U}$, and a weight function $w : \mathcal{S} \to \mathbb{Q}^+$, find a minimum weight sub-collection of $\mathcal{S}$ that includes all elements of $\mathcal{U}$.*

Definition 2.10 (Max coverage) *Given $\langle \mathcal{U}, \mathcal{S}, k \rangle$, with $\mathcal{U}$ and $\mathcal{S}$ as above, find a sub-collection of k sets that includes the maximum number of elements of $\mathcal{U}$.*

2.3 Asymmetric k-center review

The $O(\log^* n)$ algorithm of Panigrahy and Vishwanathan [66] has two phases, the *halve* phase, sometimes called the *reduce* phase, and the *augment* phase. As described above, the algorithm guesses R_{OPT} and works in the bottleneck graph $G_{R_{\mathsf{OPT}}}$. In the halve phase we find a CCV v, include it in the set of centers, mark every vertex in $\Gamma_2^+(v)$ as covered, and repeat until no CCVs remain unmarked. The CCV property ensures that, as each CCV is found and vertices are marked, the unmarked (*active*) portion of the graph can be covered with one fewer center. Hence if k'' CCVs are obtained, the active part of the graph can be covered with $k' = k - k''$ centers. The authors then prove that this unmarked portion, CCV-free, can be covered with only $k'/2$ centers if we use radius 5 instead of 1. That is to say, $k'/2$ centers suffice in the graph $G^{(5)}_{R_{\mathsf{OPT}}}$.

The k-center problem in the bottleneck graph is identical to the dominating set problem (a special case of set cover in which the sets are the Γ^+ terms). In the augment phase, the algorithm recursively uses the greedy set cover procedure. Since the optimal cover uses at most $k'/2$ centers, the first cover has size at most $\frac{k'}{2} \ln \frac{2n}{k'}$.

The centers in this first cover are themselves covered, using the greedy set cover procedure, then the centers in the second cover are covered, and so forth. After $O(\log^* n)$ iterations the algorithm finds a set of at most k' vertices that, together with the CCVs, $O(\log^* n)$-covers the unmarked portion, since the optimal solution has $k'/2$ centers. Combining these with the k'' CCVs, we have k centers covering the whole graph within $O(\log^* n)$.

Archer [4] presents two $O(\log^* k)$ algorithms, both building on the work of Panigrahy

and Vishwanathan [66]. The algorithm more directly connected with the earlier work nevertheless has two fundamental differences. Firstly, in the reduce phase Archer shows that the CCV-free portion of the graph can be covered with $2k'/3$ centers within radius 3. Secondly, he constructs a set cover-like integer program and solves the relaxation to get a total of k' *fractional* centers that cover the unmarked vertices. From these fractional centers, he obtains a 2-cover of the unmarked vertices with $k' \ln k'$ (integral) centers. These are the seed for the augment phase, which thus produces a solution with a covering radius $O(\log^* k')$ greater than the optimum. We now know that all of these approximation algorithms are asymptotically best possible [21, 42, 20].

2.4 Asymmetric weighted k-center

Recall the variant in which the costs of establishing a base or center varies according to is location. In this situation, rather than having a restriction on the *number* of centers used, each vertex has a *weight* and we have a budget W that restricts the total weight of centers used.

Definition 2.11 (Weighted k-center) *Given a weight function on the vertices, $w : V \to \mathbb{Q}^+$, and a bound $W \in \mathbb{Q}^+$, find a set $S \subseteq V$ of total weight at most W, so that S covers V with minimum radius.*

Hochbaum and Shmoys [47] gave a 3-approximation algorithm for the symmetric weighted version, applying their approach for bottleneck problems. We propose an $O(\log^* n)$ approximation for the asymmetric version that is based on Panigrahy and Vishwanathan's technique for the unweighted problem. Note that in light of the complemen-

tary hardness result [21, 42, 20], this algorithm is asymptotically the best possible. There is another variant that has both the k and the W restrictions, but we will not expand on that here.

First, a brief sketch of the algorithm, which works with bottleneck graphs. In the reduce phase, having found a CCV, v, we pick the lightest vertex u in $\Gamma^-(v)$ (which might be v itself) as a center in our solution. We then mark everything in $\Gamma_3^+(u)$ as covered, and continue looking for CCVs. We can show that there exists a 49-cover of the unmarked vertices with total weight less than a quarter of the optimum. Finally, we recursively apply a greedy procedure for weighted sets and elements $O(\log^* n)$ times, similar to the one used for set cover. The total weight of centers in our solution set is at most W.

The following lemma concerning vertex-weighted digraphs is the key to our reduce phase and is analogous to Lemma 4 in Panigrahy and Vishwanathan's paper [66] and Archer's Lemma 16 [4].

Lemma 2.1 (Cover of half the graph's weight) *Let $G = (V, E)$ be a digraph with weighted vertices, but unit edge costs. Then there is a subset $S \subseteq V$, $w(S) \leq w(V)/2$, such that every vertex with positive indegree is reachable in at most 3 steps from some vertex in S.*

Proof: To construct the set S repeat the following, to the extent possible: *Select* a vertex v with positive outdegree and if possible *select* one with indegree zero (that is, $\Upsilon^-(v)$ is empty). Compare sets $\{v\}$ and $\Upsilon^+(v)$: add the lighter set to S and remove $\Gamma^+(v)$ from G.

It is clear that the weight of S is no more than half the weight of V. We must now show that S 3-covers all non-*orphan* vertices. Since we call x a parent of y if $x \in \Upsilon^-(y)$, and thus y a child of x, y is an orphan if $\Upsilon^-(y)$ is empty.

The children of a selected vertex v, $\Upsilon^+(v)$, are clearly 1-covered. Assume v is not in S (trivial otherwise): if v was an orphan initially then ignore it. If v is an orphan when selected, but not initially, then at some previous stage in the procedure some parent of v must have been removed by the selection of a grandparent (a vertex in $\Upsilon_2^-(v)$), so v is 2-covered. Note that if one of v's parents had been selected then v would already have been removed from G.

Hence v has at least one parent when it is selected. Consequently, at that stage in the procedure, there are no vertices that have children, but are orphans, otherwise one of them would have been selected instead of v. We conclude that the sets of parents of v, $S_1 = \Upsilon^-(v)$, parents of S_1, $S_2 = \Upsilon^-(S_1)$, and parents of S_2, $S_3 = \Upsilon^-(S_2)$, are not empty. Although these sets might not be pairwise disjoint, if they contained any of v's children, then v would be 3-covered.

After v is removed, there are three possibilities for S_2: (i) Some vertex in S_3 is selected, removing part of S_2; (ii) Some vertex in S_2 is selected and removed; (iii) Some vertex in S_1 is selected, possibly making some S_2 vertices childless. One of these events *must* happen, since S_1 and S_2 are non-empty. As a consequence, v is 3-covered. ■

Recall that active vertices are those that have not yet been covered/marked. Using Lemma 2.1 we show that after removing the CCVs from the graph, we can cover the active set with half the weight of an optimum 1-cover if we are allowed to use distance 7 instead of 1.

Lemma 2.2 (Cover of half optimal weight) *Consider a subset $A \subseteq V$ that has a cover consisting of vertices of total weight W, but no CCVs. Assume there exists a set C_1 that 3-covers*

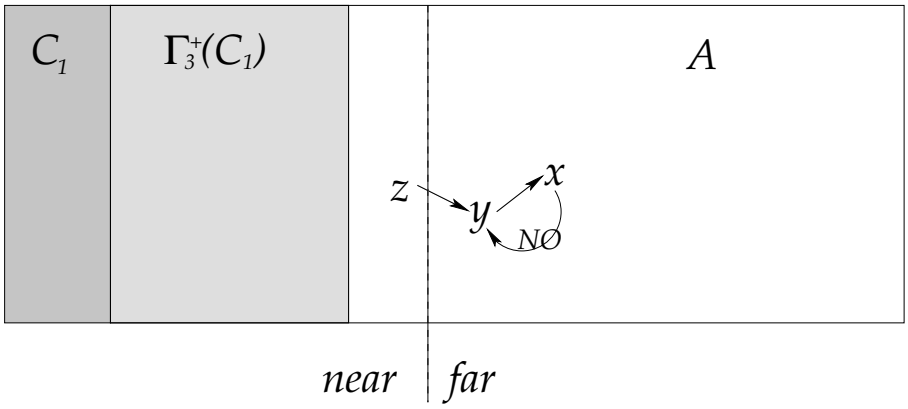

Figure 2.1: A cover with half the weight: near and far centers.

exactly $V \setminus A$. Then there exists a set of vertices S of total weight $W/2$ that, together with C_1, 7-covers A.

Proof: Let U be a subset that covers A of the optimal centers for V. We call $u \in U$ a *near* center if it can be reached in 4 steps from C_1, and a *far* center otherwise (see Figure 2.1). Since C_1 5-covers all of the nodes covered by near centers, it suffices to choose S to 6-cover the far centers, so that S will 7-cover all the nodes they cover.

Define an auxiliary graph H on the (optimal) centers U as follows. There is an edge from x to y in H if and only if x 2-covers y in G (and $x \neq y$). The idea is to show that any far center has positive indegree in H. As a result, Lemma 2.1 shows there exists a set $S \in U$ with $|S| \leq W/2$ such that S 3-covers the far centers in H, and thus 6-covers them in G.

Let x be any far center: note that $x \in A$. Since A contains no CCVs, there exists y such that y covers x, but x does not cover y. Since $x \notin \Gamma_4^+(C_1)$, $y \notin \Gamma_3^+(C_1)$, and therefore $y \in A$

(everything not 3-covered by C_1 is in A). Thus there exists a center $z \in U$, which is not x, but could be y, that covers y and therefore 2-covers x. Hence x has positive indegree in the graph H. ■

As we foreshadowed, we will use the greedy heuristic to complete the algorithm. We now analyze the performance of this heuristic in the context of the dominating set problem in node-weighted graphs. All vertices V are available as potential members of the dominating set (centers), but we need only dominate the active vertices A. The heuristic is to select the most *efficient* vertex: the one that maximizes $w(A(v))/w(v)$, where $A(v) \equiv \Gamma^+(v) \cap A$.

Lemma 2.3 (Greedy algorithm in weighted dominating set) *Let*

$$\left\langle G = (V, E), w : V \to \mathbb{Q}^+, A \subseteq V \right\rangle$$

be an instance of the dominating set problem in which a set A is to be dominated. Let w^* be the weight of an optimum solution for this instance. The greedy algorithm gives an approximation guarantee of $2 + \ln(w(A)/w^*)$.

Proof: In every application of the greedy selection there must be some vertex $v \in V$ for which

$$\frac{w(A(v))}{w(A)} \geq \frac{w(v)}{w^*} \tag{2.1}$$

otherwise no optimum solution of weight w^* would exist. This is certainly true of the most efficient vertex v, so make v a center and make all the vertices it covers inactive, leaving A' active. Now,

$$w(A') = w(A) - w(A(v)) \leq w(A)\left(1 - \frac{w(v)}{w^*}\right) < w(A)\exp\left(-\frac{w(v)}{w^*}\right).$$

After j steps of choosing the most efficient vertex, the remaining active vertices, A^j, satisfy

$$w(A^j) < w(A^0) \prod_{i=1}^{j} \exp\left(-\frac{w(v_i)}{w^*}\right) , \tag{2.2}$$

where v_i is the ith center picked (greedily) and A^0 is the original active set.

Assume that after some number of steps, say j, there are still some active elements, but the upper bound in (2.2) has just dropped below w^*. That is to say,

$$\sum_{i=1}^{j} w(v_i) > w^* \ln(w(A^0)/w^*) .$$

Before we picked the vertex v_j we had

$$\sum_{i=1}^{j-1} w(v_i) \leq w^* \ln(w(A^0)/w^*) ,$$

and so,

$$\sum_{i=1}^{j} w(v_i) \leq w^* + w^* \ln(w(A^0)/w^*) ,$$

for (2.1) tells us that $w(v_j)$ is no greater than w^*. To cover the remainder, A^j, we just use A^j itself, at a cost less than w^*. Hence the total weight of the solution is less than $w^*(2 + \ln(w(A^0)/w^*))$.

On the other hand, if the upper bound on $w(A^j)$ never drops below w^* before A^j becomes empty, then we have a solution of weight at most $w^* \ln(w(A^0)/w^*)$. ■

We now show that this tradeoff between covering radius and optimal cover size leads to an $O(\log^* n)$ approximation.

Lemma 2.4 (Recursive set cover) *Given $A \subseteq V$, such that A has a cover of weight W, and a set $C_1 \subseteq V$ that covers $V \setminus A$, we can find in polynomial time a set of vertices of total weight at most $4W$ that, together with C_1, covers A (and hence V) with a radius in $O(\log^* n)$.*

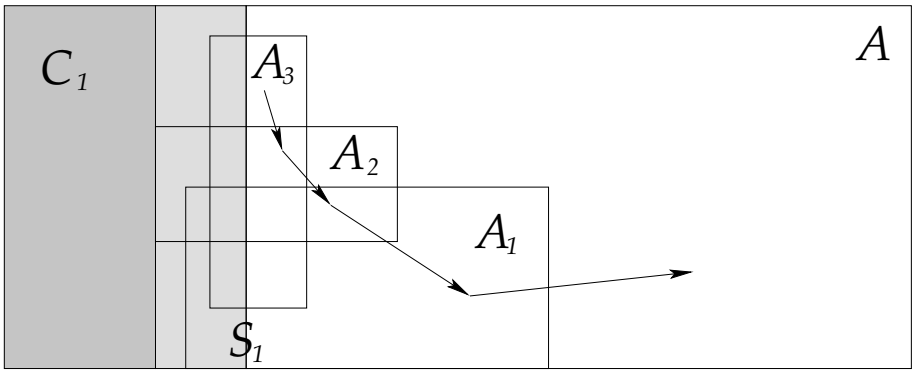

Figure 2.2: Recursive greedy set cover showing A_3 3-covering an active element

Proof: We will be applying the greedy algorithm of Lemma 2.3. The approximation guarantee is $2 + \ln(w(A)/W)$, which is less than $\log_{1.5}(w(A)/W)$ when $w(A) \geq 4W$.

Our first attempt at a solution, S_0, is all vertices of weight no more than W. Only these vertices could be in the optimum center set and their total weight is at most nW. Since C_1 covers $S_0 \setminus A$, consider $A_0 = S_0 \cap A$, which has a cover of size W. Lemma 2.3 shows that the greedy algorithm results in a set S_1 that covers A_0 and has weight

$$w(S_1) \leq W \log_{1.5}(\frac{Wn}{W}) = W \log_{1.5} n \ ,$$

assuming $n \geq 4$. As shown in Figure 2.2, the set C_1 covers $S_1 \setminus A$, so we need only consider $A_1 = S_1 \cap A$. We continue this greedy procedure and note that at the ith iteration we have $w(S_i) \leq W \log_{1.5}(w(A_{i-1})/W)$. By induction, after $O(\log^* n)$ iterations the weight of our solution set, S_i, is at most $4W$. ■

All the algorithmic tools can now be assembled to form an approximation algorithm.

Theorem 2.1 (Approximation of weighted k-center) *We can approximate the weighted k-*

center problem within factor $O(\log^* n)$ in polynomial time.

Proof: Guess the optimum radius, R_{OPT}, and work in the bottleneck graph $G_{R_{\mathsf{OPT}}}$. Initially, the active set A is V. Repeat the following as many times as possible: Pick a CCV v in A, add the lightest vertex u in $\Gamma^-(v)$ to our solution set of centers, and remove the set $\Gamma_3^+(u)$ from A. Since v is covered by an optimum center in $\Gamma^-(v)$, u is no heavier than this optimum center. Moreover, since the optimum center lies in $\Gamma^+(v)$, $\Gamma_3^+(u)$ includes everything covered by it.

Let C_1 be the centers chosen in this first phase. We know the remainder of the graph, A, has a cover of total weight $W' = W - w(C_1)$, because of our choices based on CCV and weight.

Lemma 2.2 shows that we can cover the remaining uncovered vertices with weight no more than $W'/2$ if we use covering radius 7. Applying the lemma again, we can cover the remaining vertices with weight $W'/4$ centers if we allow radius 49. So let the active set A be $V \setminus \Gamma_{49}^+(C_1)$, and recursively apply the greedy algorithm, as described in the proof of Lemma 2.4, to the graph $G_{R_{\mathsf{OPT}}}^{(49)}$. As a result, we have a set of size at most W' that covers A within radius $O(\log^* n)$. Together with C_1, this forms an $O(\log^* n)$-cover of size at most W. ■

2.5 Asymmetric p-neighbor k-center

Imagine that we wish to place k facilities so that the maximum distance of a demand point from its p^{th}-closest facility is minimized. For instance, if some of the anti-terror bases were busy, or were themselves attacked, failures in $p-1$ of them would not cause

severe blowout in response time.

Definition 2.12 (Asymmetric p-neighbor k-center) *For every subset S and vertex v in V, let $d_p(S, v)$ denote the distance from the p^{th} closest vertex in S to v. The problem is to find a subset S of at most k vertices that minimizes $\max_{v \in V \setminus S} d_p(S, v)$.*

We show that, if we allow ourselves to use $2k$ centers, we can approximate the asymmetric p-neighbor k-center problem within a factor of $O(\log^* k)$. Our algorithm is restricted to the case $p \leq n/k$, but this is reasonable as p should not be too large [54].

We use the same techniques as before, including bottleneck graphs, but in the augment phase we use the greedy algorithm for the *constrained set multicover* problem [74], instead of dominating set. That is, each element, e, needs to be covered r_e times, but each covering set can be picked at most once. The p-neighbor k-center problem has $r_e = p$ for all e. We say that an element e is *active* if it occurs in fewer than p sets chosen so far. The greedy heuristic is to pick the set that covers the most active elements. It can be shown that this algorithm achieves an approximation factor of $H_n = O(\log n)$ [74, Section 13.2]. However the following result is more appropriate to our application.

Lemma 2.5 (Greedy constrained set multicover) *Let k be the size of the optimum solution to the constrained set multicover problem. The greedy algorithm gives an approximation guarantee of $\log_{1.5}(np/k)$.*

Proof: The same kind of averaging argument used for standard set cover shows that the greedy choice of a set reduces the total number of unmarked element copies by a factor $1 - 1/k$. So after i steps, the number of copies of elements yet to be covered is

$$\begin{aligned} \text{minimize} \quad & \sum_{v \in V} y_v \\ \text{subject to} \quad & \sum_{u \in \Gamma^-(v)} y_u \geq p \qquad v \in A \\ & -y_v \geq -1 \quad v \in V \\ & y_v \geq 0 \qquad v \in V\,. \end{aligned} \tag{2.3}$$

$np(1-1/k)^i < np(e^{-1/k})^i$. Hence after $k \ln(np/k)$ steps the number of uncovered copies of elements is less than k. A naive cover of these last k element copies leads the total number of sets in the solution to be at most $k + k\ln(np/k)$. Since $p \geq 2$, this greedy algorithm has an approximation factor less than $\log_{1.5}(np/k)$. ■

If $p \leq n/k$, the approximation guarantee above is less than $\log_{1.2}(n/k)$. Applying the standard recursive approach [66], which works in the p-neighbor case, we can achieve an $O(\log^* n)$ approximation with $2k$ centers, a bicriteria result. We can lower the approximation guarantee to $O(\log^* k)$, with $2k$ centers, using Archer's LP-based priming [4], which we describe now in detail.

We first solve the LP for the constrained set multicover problem (2.3), in which y_v is the (fractional) extent to which a vertex is a center. In the solution each vertex is covered by an amount p of fractional center, out of a total of k. We can now use the greedy method to obtain an initial set of $k^2 \ln k$ centers that 2-covers every vertex in the active set with at least p centers.

Let A be the active vertices (the vertices that are covered fewer than p times) and let $A(v) = \Gamma^+(v) \cap A$. Let $y'(v) = y_v \cdot a_v$, where a_v is the number of times v still needs to be covered, and let $y'(S) = \sum_{v \in S} y'(v)$ for all subsets S. Note that $v \in A \Leftrightarrow a_v > 0$ and thus

$y'(A) = y'(V)$. The y' function indicates the extent to which an optimal fractional center is not yet covered. We will see that when the value of $y'(V)$ is low, we can be sure that we have found a reasonable cover of all the vertices.

Start with an empty set S and repeat the following until $y'(V) < p$: Choose the vertex v from $T = V - S$ maximizing $y'(\Gamma^+(v))$, add it to S, and set $a_u = a_u - 1$ for all vertices $u \in A(v)$.

Lemma 2.6 *Once $y'(V) < p$, the set S 2-covers every vertex with at least p centers.*

Proof: For every v, let $\alpha(v)$ be the active vertices amongst v and its parents, $\alpha(v) = \{u : u \in \Gamma^-(v), a_u > 0\}$, and let $\beta(v)$ be the inactive vertices in $\Gamma^-(v)$. Since $y'(V) < p$ we have

$$\sum_{u \in \alpha(v)} y_u \leq \sum_{u \in \alpha(v)} y'_u < p \ .$$

The first constraint of LP (2.3) tells us that

$$\sum_{u \in \alpha(v)} y_u + \sum_{u \in \beta(v)} y_u = \sum_{u \in \Gamma^-(v)} y_u \geq p \ ,$$

and thus $\sum_{u \in \beta(v)} y_u > 0$. We conclude that there must be at least one vertex in $\beta(v)$; the p vertices covering this vertex 2-cover v. ■

The following lemma corresponds to Archer's Lemma 4 [4].

Lemma 2.7 *There exists $v \in T$ such that*

$$y'(A(v)) \geq \frac{y'(A)}{y(T)} \ .$$

Proof: We take a weighted average of $y'(A(v))$ over $v \in T$.

$$\begin{aligned}\frac{1}{y(T)}\sum_{v\in T} y_v \cdot y'(A(v)) &= \frac{1}{y(T)}\sum_{v\in T}\sum_{u\in A(v)} y_v \cdot y'(u) \\ &= \frac{1}{y(T)}\sum_{u\in A} y'(u) \sum_{v\in \Gamma^-(u)\cap T} y_v \\ &\geq \frac{1}{y(T)}\sum_{u\in A} y'(u)\end{aligned}$$

The inequality follows from the fact that for all $u \in A, y'(u) \geq 0$ and $y(\Gamma^-(u) \cap T) \geq 1$ (otherwise there would be more than $p-1$ members of $\Gamma^-(u)$ in S). Since some term is at least as large as the weighted average, the lemma follows. ■

Lemma 2.8

$$|S| \leq k^2 \ln k \ .$$

Proof: Due to Lemma 2.7, the vertex v chosen in every application of the greedy method has $y'(\Gamma^+(v)) = y'(A(v)) \geq y'(A)/y(T)$. In this proof we focus on one iteration at a time and let A' stand for the active vertices *after* the iteration and A for those before. Now,

$$\begin{aligned}y'(A') &= y'(A) - y(A(v)) \\ &\leq y'(A) - y'(A(v))/p\,,\end{aligned}$$

as $y(B) \geq y'(B)/p$ for any set B. Since we chose greedily, this is less than or equal to

$$\begin{aligned}y'(A) - \frac{y'(A)}{y(T)\cdot p} &\leq y'(A) - \frac{y'(A)}{kp} \\ &= y'(V)(1-\frac{1}{kp}) \\ &< y'(V)\exp(-\frac{1}{kp})\,,\end{aligned}$$

as $y(T) \leq k$. Initially, $y'(V) = kp$, so $y'(V)$ will be less than p after at most $kp \ln(k)$ iterations. By definition, $p \leq k$, so we have $|S| \leq k^2 \ln k$. ■

Repeatedly applying the greedy procedure for constrained set multicover, this time for $O(\log^* k)$ iterations, we get $2k$ centers that cover all active vertices within $O(\log^* k)$.

2.6 Inapproximability results

In this section we provide inapproximability results for the asymmetric versions of the k-center with outliers, priority k-center, and k-supplier problems. These problems all admit constant factor approximation algorithms in the symmetric case (Table 2.1).

Asymmetric k-center with outliers

A disadvantage of the standard k-center problem is that a few distant clients can force centers to be located in isolated places. This situation is averted in the following variant problem, in which a small subset of clients may be denied service, and some points are forbidden from being centers.

Definition 2.13 (k-center with outliers and forbidden centers) *Find a set $S \subseteq C$, where C is the set of vertices allowed to be centers, such that $|S| \leq k$ and S covers at least p nodes, with minimum radius.*

Theorem 2.2 *For any function $\alpha(n)$, the asymmetric k-center problem with outliers (and forbidden centers) cannot be approximated within a factor of $\alpha(n)$ in polynomial time, unless $P = NP$.*

Proof: We reduce instance $\langle U, \mathcal{S}, k\rangle$ of max coverage to our problem. Construct vertex sets A and B so that for each set $S \in \mathcal{S}$ there is $v_S \in A$, and for each element $e \in U$ there is $v_e \in B$. From each vertex $v_S \in A$, create an edge of unit length to vertex $v_e \in B$ if $e \in S$. Let $p = |B| + k$.

If the optimum value of the max coverage instance is $|\mathcal{U}|$, then the k nodes in A that correspond to some optimal sub-collection will cover p nodes within radius 1. An $\alpha(n)$-approximation algorithm will thus return k centers that cover p nodes in some *finite* distance. If the maximum coverage with k sets is less than $|\mathcal{U}|$, then the optimum covering radius for p nodes, using k centers, is infinite. Since our approximation algorithm can distinguish between these two cases, the approximation problem must be NP-complete.

■

Note that the proof never relied on the fact that the B vertices were forbidden from being centers—setting p to $|B| + k$ ensured this.

Asymmetric priority k-center

Perhaps some demand points need centers closer to them than other demand points. This situation is captured by the priority k-center problem, in which the distance to a demand vertex is effectively enlarged by its priority. Note that the triangle inequality still holds for the original distances.

Definition 2.14 (Priority k-center) *Given a priority function $p : V \to \mathbb{Q}^+$ on the vertices, find $S \subseteq V$, $|S| \leq k$, that minimizes R so that for every $v \in V$ there exists a center $c \in S$ for which $p_v d_{cv} \leq R$.*

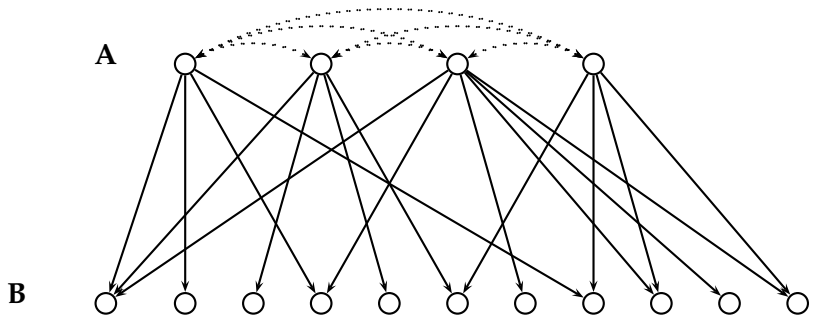

Figure 2.3: Reduction from max coverage to asymmetric priority k-center. Solid lines have length 1, dotted lines length ℓ.

Theorem 2.3 *For any polynomial time computable function $\alpha(n)$, the asymmetric k-center problem with priorities cannot be approximated within a factor of $\alpha(n)$ in polynomial time, unless $P = NP$.*

Proof: The construction of the sets A and B is similar to the proof of Theorem 2.2. Again, we have the unit length set-element edges from A to B, but this time we make the subgraph on A a complete digraph, with every edge of length ℓ, as in Figure 2.3. The nodes in set A have priority 1 and the nodes in set B priority ℓ.

If there exists a collection of k sets that cover all elements, then there exist k vertices in A that cover every vertex in A and B within radius ℓ. If there do not exist k such sets, then the optimal covering radius using k centers is $\ell^2 + \ell$: some vertex in B is at distance $\ell + 1$ from its nearest center and has priority ℓ. If we set ℓ to $\alpha(n)$, our algorithm can distinguish between the two types of max coverage instance. Therefore the approximation problem is NP-complete. ■

Asymmetric k-supplier

In the k-supplier problem the vertex set is segregated into suppliers and customers. Only supplier vertices can be centers and only customer vertices need to be covered.

Definition 2.15 (k-supplier) *Given a set of suppliers Σ and a set of customers C, with Σ and C disjoint, find a subset $S \subseteq \Sigma$ that minimizes R such that S covers C within R.*

Theorem 2.4 *For any function $\alpha(n)$, the asymmetric k-supplier problem cannot be approximated within a factor of $\alpha(n)$ in polynomial time, unless $P = NP$.*

Proof: We use a reduction from max coverage similar to the proof of Theorem 2.2. ■

2.7 Pattern rotation

In this final section, we divert our attention somewhat to look at an elegant combinatorial covering problem related to the asymmetric k-center problem.

Before the announcement of the $\Omega(\log^* n)$ hardness of approximation result [21, 42, 20], we considered a number of graph constructions that could have led to a proof of a super-constant integrality gap for asymmetric k-center. In particular, we considered graphs that were regular in the following way. Let the vertices be labeled $0, 1, \ldots, n-1$ and denote the set of vertices that 0 covers by S; the vertex i has edges only to those vertices of the form $i + s \mod n$ for s in S, except i itself. See Figure 2.4 for an example of this type of regular graph.

Imagine a covering of these n vertices using fractional centers. It is easy to see that the solution with $y_i = 1/|S|$ for all i is an optimal fractional covering. We were interested to

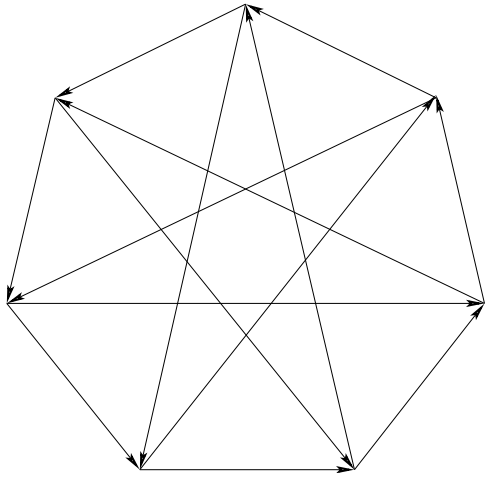

Figure 2.4: Simple pattern rotation instance

find the minimum covering radius so that $n/|S|$ (integral) centers would be sufficient to cover all the vertices. We had hoped to find a family of graphs (and subsets S) in which these integral covering radiuses grew with n, thus showing a super-constant integrality gap for the asymmetric k-center problem. Despite not making significant progress in this direction, we did prove that a related problem has a logarithmic integrality gap.

In the asymmetric k-center problem, we fix the size of the cover and try to minimize the covering radius. We could, as is generally the case with set cover-type problems, insist on a unit radius and try to find the minimum size cover. We now describe an abstraction of this problem, which we call *pattern rotation*.

Let S be a subset of $\mathbb{Z}_n$ and call the set

$$S + a = \{s + a \mod n \mid s \in S\}$$

a rotation of S In an abuse of notation we will sometimes refer to a as the rotation. We

consider the problem of covering $\mathbb{Z}_n$ with rotations of (the *pattern*) S. The covering number of S is the size of the smallest set A such that $S+A = \mathbb{Z}_n$, where $S+A = \bigcup_{a\in A}(S+a)$. We extend this idea to fractional coverings, in which each element i is given a weight y_i and the sum of the y_i such that i covers j ($j \in S + i$), must be at least 1.

Not much seems to be known about the problem of finding the (integral) covering number of a given set, or indeed a minimal set of rotations. For instance, it may not be known whether the problem is NP-hard. Moreover, even if we had an oracle for the decision problem, *Is the covering number of S less than x?*, it is not clear how we could use this to produce an optimal family of rotations. That is to say, we assume without loss of generality that $0 \in A$, but if the oracle were to tell us that, say, 3 were in A, we would not know what to ask the oracle next. Unlike set cover, in which removing a set and the elements it covers results in another (smaller) set cover instance, removing $S + 3$ from $\mathbb{Z}_n$ would result in a new kind of problem instance, for which the oracle would not be of use. Despite these uncertainties, we can be sure that an approximation algorithm based on the canonical LP will not have a performance guarantee in $o(\log n)$—the greedy set cover heuristic provides a $\Theta(\log n)$ approximation.

Lemma 2.9 *There exists some collection of sets $\{S_n\}$ for which the ratio between the (integral) and fractional covering numbers is in $\Omega(\log n)$.*

Proof: For each n, create a set S_n by selecting each element of $\mathbb{Z}_n$ independently with probability $1/2$. The Chernoff bound tells us that the probability that the size of S_n is less than $n/4$, implying its fractional covering number is > 4, is at most $e^{-n/32}$.

Consider a family of rotations $A = \{a_0, a_1, \ldots, a_{k-1}\}$, with $a_0 = 0$. The probability

that a particular element $x \in \mathbb{Z}_n$ is not covered by any rotation $S_n + a_i$ is exactly $1/2^k$. Let us say that two elements x and y of $\mathbb{Z}_n$ are independent if there is no pair i, j for which $x - a_i = y - a_j$. That is to say, there is no other element of $\mathbb{Z}_n$ whose (non)inclusion in S_n implies the (non)inclusion of both x and y. For any x there are only $k^2 - k$ elements of $\mathbb{Z}_n$ that are not independent of it. Given a set of rotations A there is therefore a set X_A of n/k^2 elements that are mutually independent. The probability that all elements of X_A are covered is

$$\left(1 - \frac{1}{2^k}\right)^{n/k^2} \leq \exp\left(-\frac{n}{2^k k^2}\right).$$

Now, for a fixed k there are $\binom{n}{k} \leq n^k$ sets of k rotations, each of which has a corresponding set X_A of mutually independent elements. The probability that all these sets X_A are covered is at most

$$n^k \exp\left(-\frac{n}{2^k k^2}\right),$$

which, if $k \leq (\log n)/2$, is less than $1/2$ for $n > 2^{34}$. Therefore, with probability at least $1/2$ no set of k rotations can cover all of $\mathbb{Z}_n$ (since some X_A) set is uncovered. That is to say, there exists some pattern S_n for which this property holds, and thus has integral covering number greater than $(\log n)/2$, but fractional covering number at most 4. ■

Chapter 3

Correlation clustering

3.1 Introduction

Correlation clustering, introduced to the theoretical computer science community by Bansal, Blum and Chawla [7], departs from the traditional distance paradigm of clustering. All we have at our disposal is *qualitative* information from a judge: a labeling of each pair of elements as either similar or dissimilar. These judgments could, for example, be based on comparisons of records in a database. We are not provided with any quantitative distance information about the pairs. Our aim is to produce a partitioning into clusters that puts similar objects in the same cluster and dissimilar objects in different clusters, to the maximum extent possible. If there exists a clustering that is *correct* for every edge, then the problem is trivially solved by identifying as clusters the connected components in the graph of similar pairs. When the judge has made mistakes, interesting and non-trivial questions arise: primarily, finding a clustering that differs from the judge's verdicts on the fewest possible pairs. Bansal *et al.* pointed out that correla-

tion clustering corresponds to agnostic learning [52], when viewed as a machine learning problem. The edge labels are the examples and we are only allowed to use partitionings as hypotheses for the target function.

An obvious graph-theoretic formulation of the problem is the following: given a graph $G = (V, E)$ with each edge labeled either "$+$" (similar) or "$-$" (dissimilar), find a partitioning of the vertices into clusters that agrees as much as possible with the edge labels. If every pair of elements is labeled either $+$ or $-$, then G will be a complete graph. In order to capture situations where the judge might be unable to tell whether certain pairs of elements are similar or dissimilar, we do not insist on the input being a complete graph. One upshot of the clustering of an incomplete instance is the deduction of the missing labels from the existing ones.

Bansal *et al.* proposed three specific correlation clustering problems that are equivalent from an optimization point of view, but rather different in their approximation properties. An edge is an *agreement* if it is a $+$ edge within a cluster, or a $-$ edge across two distinct clusters; a *disagreement* is a $+$ edge between clusters or a $-$ edge within a cluster. The MAXAGREE problem seeks to maximize the number of agreements, the MINDISAGREE problem seeks to minimize the number of disagreements, while the MAXCORR problem asks us to maximize the correlation: the number of agreements minus the number of disagreements. Note that we do not need to specify the number of clusters k as a parameter. We have only a single objective; whether the optimal solution uses few or many clusters is automatically dictated by the edge labels.

In some instances, the judge might provide confidence information for each of the labels. This is captured by assigning *weights* to the edges; one can then consider natural

weighted versions of MAXAGREE, MINDISAGREE, and MAXCORR.

3.2 Previous and related work

Correlation clustering on complete graphs seems to have been first considered by Ben-Dor *et al.* [10] motivated by some computational biology questions. Later, Shamir *et al.* [71] studied the computational complexity of the problem and showed that MAXAGREE (and hence also MINDISAGREE and MAXCORR) is NP-hard for complete graphs. Shamir *et al.* used the term *Cluster editing* to refer to this problem; recent algorithms for fixed parameter versions are presented by Gramm *et al.* [39]. Independently, Chen *et al.* [19] examined a very similar problem in the context of phylogeny trees, essentially showing that MINDISAGREE is NP-hard.

As mentioned earlier, Bansal, Blum, and Chawla [7] considered this problem independently. They initiated the study of approximate solutions to MINDISAGREE, MAXAGREE, and MAXCORR, focusing mainly on the case when G is complete. Bansal *et al.* gave a polynomial time approximation scheme (PTAS) for MAXAGREE on complete graphs. For the minimization version MINDISAGREE, they gave an approximation algorithm with constant performance ratio. The constant is a rather large one, so it should be viewed as a qualitative result, demonstrating that a constant factor approximation can be achieved. In the full version of their work [8], Bansal *et al.* provide a simple algorithm that is at most a factor three worse than the best partitioning into *two* clusters. The only observation they made regarding MAXCORR is that the optimal solution has correlation in $\Omega(n)$ for complete instances.

Bansal *et al.* also posed several open questions including those of demonstrating hardness of approximation results for complete graphs and understanding the problem on general graphs. These questions motivated a number of groups to work on this problem simultaneously.

Both Demaine and Immorlica [25], and Emanuel and Fiat [28], independently from each other and from the research presented in the following chapters, announced results on clustering with qualitative information. These two papers focus on MINDISAGREE in general graphs. Demaine and Immorlica [25] present a factor $O(\log n)$ algorithm for general graphs, based on *region growing*, and demonstrate an approximation preserving reduction from (weighted) minimum multicut. They also provide an $O(r^3)$ approximation algorithm for MINDISAGREE in $K_{r,r}$-minor-free graphs. Emanuel and Fiat present reductions both to and from minimum multicut; in particular, the authors show a reduction from unweighted multicut to unweighted MINDISAGREE. For MAXAGREE on general graphs, Swamy [72], again independently from the research here, provided a factor 0.7666 approximation algorithm (very slightly better than the factor we present in Chapter 5).

3.3 Our results

We have answered several questions left open by the work of Bansal *et al.* [7]. Table 3.1 indicates how our results provide a better overview of the approximability of the various flavors of correlation clustering.

Table 3.1: Compendium of principal correlation clustering approximation results, with algorithmic factors above the lines and hardness bounds below. Results presented in this dissertation are in boldface.

Minimizing disagreements

Complete		General	
4	[14]	$\mathbf{O(\log n)}$	[14, 28, 25]
17433	[7]		
$\mathbf{c > 1}$	[14]	$\mathbf{29/28} - \varepsilon$	[14]
		min multicut	[14, 28, 25]
		$c > 1$	[8]

Maximizing agreements

Complete		General	
PTAS	[7]	0.7666	[72]
		0.7664	[14]
		$\mathbf{115/116} + \varepsilon$	[14]

Maximizing correlation

		General	
		$\mathbf{\Omega(1/\log n)}$	[16]
		$\mathbf{43/44} + \varepsilon$	[16]

Complete graphs

Our main algorithmic result is a factor 4 approximation algorithm ALGCOMPLETE for MINDISAGREE on complete graphs. This significantly improves on the performance ratio of Bansal *et al.*'s combinatorial algorithm [7]. Our algorithm is based on a natural linear programming relaxation; it rounds the fractional solution (a semi-metric on the vertices) using the *region growing* approach. The completeness of the graph allows us to to achieve a constant approximation using region growing, instead of the usual logarithmic factor [35]. The integrality gap of our LP formulation is 2 and we also show that beating factor 3 would require significant departure from our strategy. To complement our algorithmic result, we also prove that MINDISAGREE on complete graphs is APX-hard (that is, NP-hard to approximate within some constant factor greater than 1) via a somewhat intricate reduction. The reduction that Bansal *et al.* use to prove NP-hardness does not yield APX-hardness. In contrast, MAXAGREE does admit a PTAS on complete graphs [7].

General graphs

Bansal *et al.* did not give any algorithms for general graphs, but noted that MINDISAGREE is APX-hard [8]. They provided evidence that MAXAGREE is unlikely to admit a PTAS (unlike the complete graph case) by showing that a PTAS would imply a much better algorithm for coloring 3-colorable graphs than is currently known. We give a factor $O(\log n)$ approximation algorithm for MINDISAGREE—this follows from a straightforward modification of the Garg, Vazirani, Yannakakis (GVY) region growing algorithm

for minimum multicut [35]. We also note that MINDISAGREE is at least as hard to approximate as multicut, so a constant factor approximation algorithm would be a major breakthrough.

We prove that MAXAGREE is APX-hard and thereby provide a concrete hardness result—in contrast to the above *evidence* of hardness based on a relation to graph coloring. Complementary hardness results follow for MINDISAGREE and MAXCORR. On the algorithmic side, the naive $1/2$-approximation algorithm, namely choosing the better of placing all elements in a single cluster and placing each of them in a separate cluster, was the best known for MAXAGREE. We note in passing that this implies that the optimum value of MAXCORR is always nonnegative. We give a factor 0.7664 approximation algorithm Best(H_2, H_3) based on rounding a semidefinite programming relaxation. Moreover, if there exists a clustering that correctly classifies most of the edges, then our algorithm will also find one with a similar property (we defer the quantitative statement to the relevant technical section). Our interest in the latter result is due in part to the fact that it brings out some of the difficulty that must be overcome if one tries to prove a super-constant factor inapproximability result for MINDISAGREE. Such a result would have to focus on instances where an almost perfect clustering exists for both the *yes* and *no* cases of the gap reduction.

We provide the first approximation algorithm *ApproxMaxCorr* for maximizing the correlation. Intuitively, this problem seems harder than MAXAGREE, as the optimum value may be very close to zero. The MAXCORR problem restricted to two clusters is a special case of a type of quadratic program for which we have an SDP-based $\Omega(1/\log n)$ approximation. We show that taking the better of the singleton-clusters solution and the best

two-cluster solution provides a $1/3$ approximation for the general MAXCORR problem. Therefore, we obtain an $\Omega(1/\log n)$ approximation for MAXCORR.

The algorithms for MINDISAGREE, MAXAGREE, and MAXCORR are in Chapters 4, 5, and 6, respectively. All of the hardness of approximation proofs follow in Chapter 7. Finally, in Chapter 8 we apply some well-known techniques to reduce the running time of some of our approximation algorithms.

Chapter 4

Minimizing disagreements

4.1 General graphs

We describe a natural LP relaxation for MINDISAGREE. This is very similar to the LP used in the GVY minimum multicut algorithm [35].

A partitioning into clusters can be represented with a set of binary variables, one for each pair of vertices. If i and j are in the same cluster then x_{ij} is 0, if they are in different clusters then x_{ij} is 1. Since each cluster is an equivalence class, we know that if $x_{ij} = 0$ and $x_{jk} = 0$, then $x_{ik} = 0$. We can express this fact using the triangle inequality,

$$x_{ik} \leq x_{ij} + x_{jk} \, .$$

The objective is to minimize the number of mistakes: the number of positive edges for which x_{ij} is one and the number of negative edges for which x_{ij} is zero. The integer program (4.1) summarizes the situation: $+(ij)$ indicates that the edge between i and j has a positive label, while $-(ij)$ indicates a negative label. The confidence that the judge

$$\begin{array}{lll} \text{minimize} & \sum_{+(ij)} w_{ij} \cdot x_{ij} + \sum_{-(ij)} w_{ij} \cdot (1 - x_{ij}) & \\ \text{subject to} & x_{ik} \leq x_{ij} + x_{jk} & \text{for all } i, j, k \\ & x_{ij} \in \{0, 1\} & \text{for all } i, j \end{array} \tag{4.1}$$

places on the (dis)similarity label between i and j is represented by the weight w_{ij}. The LP relaxation is obtained by replacing the integer constraints in (4.1) with $0 \leq x_{ij} \leq 1$ for all i, j.

Let the value of the optimal LP solution be denoted by $\mathsf{OPT}_{\mathsf{LP}}$. A fairly straightforward application of the GVY region growing procedure yields a solution of cost at most $O(\log n)\mathsf{OPT}_{\mathsf{LP}}$. We briefly describe this algorithm, ALGGENERAL, and outline its analysis.

We will refer to x_{ij} as the *distance* between i and j, which is consistent with the fact that x_{ij} is a semi-metric in the range $[0, 1]$. Intuitively, points that are *close* should be placed in the same cluster and points that are *far* should be placed in different clusters. Let $B_x(i, r)$ denote the set of points whose distance from i is less than or equal to r. For a set of vertices S, let $\delta(S)$ be the set of edges between S and $\overline{S}$.

Theorem 4.1 ALGGENERAL *achieves an* $O(\log n)$ *approximation for* MINDISAGREE *on general graphs.*

Proof: The GVY region growing procedure suggests the choice of radius r in step 2(a) of the algorithm. Set $V_x^+(i, r)$ to be

$$\frac{\mathsf{OPT}_{\mathsf{LP}}}{n} + \sum_{+(uv) \in B_x(i,r)} w_{uv} x_{uv} + \sum_{+(uv) \in \delta(B_x(i,r))} w_{uv}(r - x_{iu}) \,.$$

ALGGENERAL

1. $\mathcal{C} \leftarrow \emptyset$. /* Collection of clusters */

2. While there exist i, j in the graph such that $x_{ij} > 2/3$:

   (a) Let $S = B_x(i, r)$ for some $r < 1/3$. /* See proof for value of r */

   (b) $\mathcal{C} \leftarrow \mathcal{C} \cup \{S\}$.

   (c) Remove S and $\delta(S)$ from the current graph.

3. Return $\mathcal{C}$.

This is the contribution to the LP solution from positive edges that have at least one endpoint in $B_x(i, r)$, plus an additional amount $\mathsf{OPT}_{\mathsf{LP}}/n$. Let $W_x^+(i, r)$ denote the sum of weights of positive edges in $\delta(B_x(i, r))$. We choose $r < 1/3$ so that the ratio of $W_x^+(i, r)$ to $V_x^+(i, r)$ is minimized. The analysis technique in [35] can be used to show that there exists a radius $r < 1/3$ such that $W_x^+(i, r) \leq (3 \log n) V_x^+(i, r)$. This and the triangle inequality imply that the total weight of positive edges with end points in different clusters is in $O(\log n)\mathsf{OPT}_{\mathsf{LP}}$.

Now we account for the negative edges. Any negative edge ij that ends up inside a cluster in our solution contributes $w_{ij} \cdot (1 - x_{ij})$ to the LP, which is at least $w_{ij}/3$, since $x_{ij} \leq 2/3$. On the other hand, we pay w_{ij} for this edge. This implies that the total weight of negative edges with end points in the same cluster is at most $O(\log n)\mathsf{OPT}_{\mathsf{LP}}$. ■

The $O(\log n)$ approximation ratio we obtain from our LP is asymptotically the best possible. Our LP formulation has integrality gap $\Omega(\log n)$, as shown by examples similar

$$\begin{aligned} \text{minimize} \quad & \sum_{+(ij)} x_{ij} + \sum_{-(ij)} (1 - x_{ij}) \\ \text{subject to} \quad & x_{ik} \leq x_{ij} + x_{jk} \quad \text{for all } i, j, k \\ & 0 \leq x_{ij} \leq 1 \quad \text{for all } i, j \end{aligned} \tag{4.2}$$

to the expander gap examples for minimum multicut [35].

We expect that a procedure such as this one, which *learns* distances from similarity judgment information, will have further applications in situations where no natural distance function exists.

4.2 Complete graphs

We present a factor four algorithm for minimizing disagreements in the complete graph. In contrast to Bansal *et al.* [7], who devised a combinatorial algorithm with factor 17433, our algorithm uses a linear programming formulation of the problem.

Our approach bears some similarity to ALGGENERAL in Section 4.1. Once the linear relaxation (4.2) of the program for the is solved, in polynomial time, we are ready for our factor four approximation algorithm.

We refer to x_{ij} not only as the *distance* between i and j, but also as the *length* of edge ij. The procedure we present, ALGCOMPLETE, illustrated also in Figure 4.1, clearly describes a partitioning. We analyze its performance by comparing the number of mistakes incurred to the LP costs of appropriate edges.

Let us reflect on the natural intuition behind the algorithm. Intuitively, the LP solution x_{ui} gives a handle on how different u and i are: the smaller the value of x_{ui} the more

ALGCOMPLETE

1. Let $S = V$ and repeat the following steps until S is empty.

2. Select a vertex u arbitrarily from S.

3. Let T be the set of vertices whose distance from u is no greater than $1/2$,

   except u itself: $B_x(u, 1/2) - \{u\}$.

4. If the average distance of the vertices in T from u is not less than $1/4$,

   then make $C = \{u\}$ a singleton cluster and jump to step 6.

5. If the average distance is less than $1/4$, then make $C = \{u\} \cup T$ a cluster.

6. Let $S = S - C$ and jump to step 2 (the start of the loop).

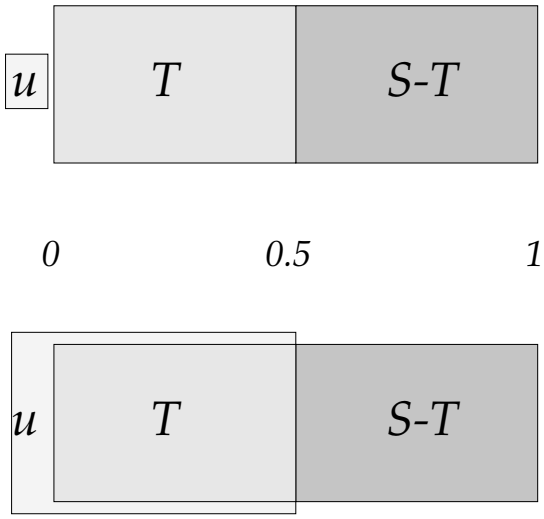

Figure 4.1: Illustration of the two main choices in ALGCOMPLETE: numerical annotations are the *distances* from u

incentive there is to place u and i in the same cluster. Therefore, it makes sense to cluster the points close to u (in a ball $B_x(u, r)$) in one cluster, say C, together with u. If both i and j are close to u, but are connected by a negative edge, we will cluster them together and make a mistake, but the LP cost of that edge $1 - x_{ij}$ will also be high since $x_{ij} \leq x_{iu} + x_{ju}$ must also be small. This basic strategy works well with negative edges. However, there is a problem if most of the vertices in C are near its *periphery*, that is, at distance close to r from u. In such a case, the LP might have very low cost x_{ij} for some $+(ij)$ crossing the cut, compared to the unit cost that the algorithm incurs on the same edge. A natural measure of whether this phenomenon could occur is the average distance from u of points in C. If this is large, then there could be many points on the periphery, and the above difficulty could occur, so we simply place u in its own cluster. It turns out, from the analysis that follows, that the best criterion for choosing between the ball cluster and a singleton cluster, is whether the average distance is greater or less than $1/4$.

At each iteration of the loop, we relabel the vertices (other than u) so that $i < j$ if $x_{ui} < x_{uj}$, breaking ties arbitrarily. The triangle inequality tells us that for $i < j$,

$$x_{uj} \leq x_{ui} + x_{ij} \quad \text{and} \quad x_{ij} \leq x_{ui} + x_{uj}\,.$$

Observation 4.1 *The LP cost of a positive edge ij, x_{ij}, is at least $x_{uj} - x_{ui}$. The LP cost of a negative edge ij, $1 - x_{ij}$, is at least $\max\{0, 1 - x_{ui} - x_{uj}\}$.*

Associated with the new cluster, C, are the edges within C and the edges between C and $S - C$. We show that the mistakes in each iteration of ALGCOMPLETE can be charged to the LP costs of the edges associated with the new cluster C. Let us now consider one iteration at a time, starting with the case when a singleton cluster is formed.

Singleton cluster

The edges associated with a singleton cluster are simply all the edges incident to u: the positive ones are the mistakes. We know from our choice in step 4 that

$$\sum_{i \in T} x_{ui} \geq |T|/4 \,.$$

For $i \in T$, $1 - x_{ui} \geq x_{ui}$, so the LP cost of *all* edges from u to T, is at least $|T|/4$. The number of (positive) edge mistakes from u to T, which is at most $|T|$, is thus at most four times the LP cost of edges from u to T.

The remaining edges associated with this cluster are between u and $S - T$. Each positive mistake incident on u has distance, and thus LP cost, greater than $1/2$; so the number of mistakes is at most twice the LP cost of these edges.

Cluster with T

We now turn to the case in which $C = \{u\} \cup T$. There are two kinds of mistakes in this case: negative edges inside C and positive edges between C and $S - C$.

(i) Negative edge mistakes

If both i and j are within distance $3/8$ of u, then the LP cost of negative edge ij is at least $1/4$, by Observation 4.1. This accounts for the mistake within factor 4.

Each remaining negative edge mistake ij will be charged to vertex j, the vertex that is further from u (see Figure 4.2). So *fix* j and assume x_{uj} lies in the range $(3/8, 1/2]$. Observation 4.1 tells us that the total LP cost of all the edges within C, associated with j,

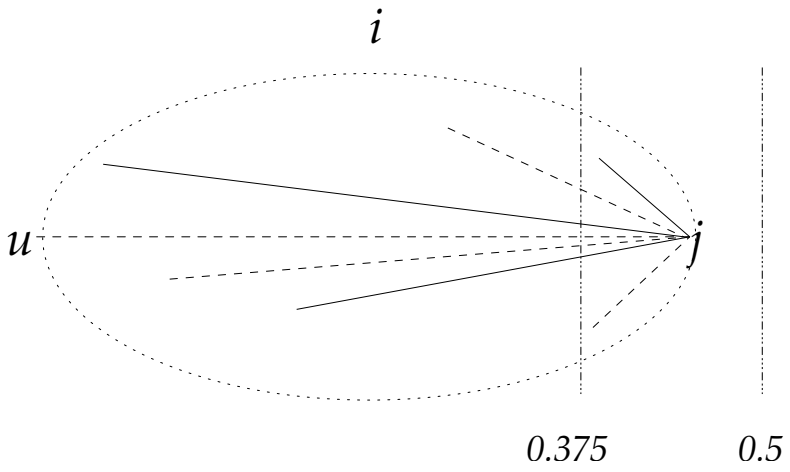

Figure 4.2: Charging mistakes and LP costs to the further (fixed) vertex j

is at least

$$\sum_{i:i<j,+(ij)} (x_{uj} - x_{ui}) + \sum_{i:i<j,-(ij)} (1 - x_{ui} - x_{uj}) .$$

We let $x_{vv} = 0$ for all v so that this summation is well-defined. Denote by p_j the number of positive edges ij for which $i < j$, and let n_j stand for the number of such negative edges. The total cost is then

$$p_j x_{uj} + n_j(1 - x_{uj}) - \sum_{i:i<j} x_{ui} . \tag{4.3}$$

Since we are including T in C, we know that the average value of x_{ui} is less than $1/4$ for $i \in T$. The summation above is over the set $\{i : i < j\}$, but since $x_{ui} \geq 3/8$ for $i > j$, the average value of the summation terms in (4.3) is less than $1/4$. Hence the LP cost is greater than

$$p_j x_{uj} + n_j(1 - x_{uj}) - \frac{p_j + n_j}{4} . \tag{4.4}$$

The number of mistakes associated with j is merely n_j. The LP cost is bounded below

by a linear function (4.4) that ranges from $p_j/8 + 3n_j/8$, when $x_{uj} = 3/8$, to $p_j/4 + n_j/4$, when $x_{uj} = 1/2$. Therefore the LP cost is at least $n_j/4$ and all the (negative) mistakes are accounted for within factor four. Since this property holds for every j in the range $(3/8, 1/2]$, we conclude that the total number of negative edge mistakes is accounted for by appropriate LP edge costs within factor four.

(ii) Positive edge mistakes

Consider positive edges ij that cross the distance $1/2$ boundary: $x_{ui} \leq 1/2$, but $x_{uj} > 1/2$. In particular, if $x_{uj} \geq 3/4$, then $x_{uj} - x_{ui} \geq 1/4$ and so each such positive edge pays for itself within factor four.

Again, we associate each remaining edge with the vertex that is further from u. So fix j and assume that x_{uj} is in the range $(1/2, 3/4)$. The LP cost of the edges associated with j is

$$p_j x_{uj} + n_j(1 - x_{uj}) - \sum_{i \in T \cup \{u\}} x_{ui}\,,$$

which is strictly greater than (4.4). This time, the linear function lower bound ranges between $p_j/4 + n_j/4$, when $x_{uj} = 1/2$, and $p_j/2$, when $x_{uj} = 3/4$. The number of (positive) mistakes is p_j so again we can pay for these within factor 4 of the LP cost. This argument holds for all j and thus for all positive edge mistakes.

Summary

Each choice of cluster leads to a ratio of at most four between the number of mistakes and the linear programming cost of associated edges. Since in past iterations we never

charged to edges within S, and in future iterations we charge only to edges within $S - C$, we have a factor four approximation algorithm.

Theorem 4.2 ALGCOMPLETE *achieves a factor* 4 *approximation for* MINDISAGREE *on complete graphs.*

As we remarked earlier, if we assume that all positive edges are correct, the problem is trivial as it reduces to finding connected components. Shamir, Sharan, and Tsur [71] studied the *cluster deletion* problem, in which all *negative* edges are deemed to be correct and must be cut, and showed it to be APX-hard. In this case, the problem analogous to MINDISAGREE is to find a clustering with the fewest possible positive edges crossing cluster boundaries. Our algorithm for MINDISAGREE also achieves a 4 approximation in this variant. The idea is to add the constraints $x_{ij} = 1$ in the linear program for each $-(ij)$, and then run ALGCOMPLETE on the LP solution. We make the minor amendment, which does not affect the proof of Theorem 4.2 substantially, that T does not include the vertices whose distance from u is exactly $1/2$. Thus each cluster C has diameter less than 1 and the endpoints of a negative edge are never placed in the same cluster. The analysis for the number of mistakes on positive edges remains identical. With this variant, as with MINDISAGREE, it is an interesting question whether the factor 4 can be improved.

4.3 Approximation limitations

Integrality gap

Any approximation technique that is based on the linear program (4.2) is limited by its integrality gap. The following *star* example, in Figure 4.3, shows this gap is at least two.

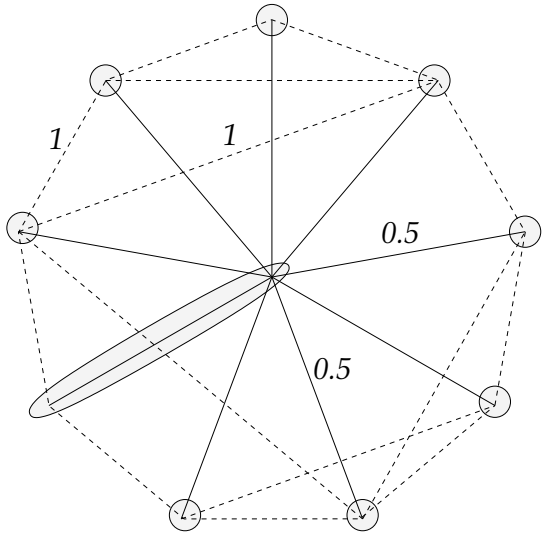

Figure 4.3: MINDISAGREE instance with integrality gap almost 2, showing both the fractional optimum (with distances) and integral optimum (with clusters). Some edges have been omitted for clarity.

Place n vertices around a single center vertex so that the center is joined to the others with positive edges, but the perimeter vertices have negative edges between them. In an optimum fractional solution the positive edges have length $1/2$ and the negative edges have length 1, so $\mathsf{OPT}_{\mathsf{LP}} = n/2$. An optimal clustering places all the perimeter vertices in singleton clusters, except for one, which is in a cluster with the center, so $\mathsf{OPT} = n - 1$. The gap, $2(n-1)/n$, has limit 2 as n increases.

Limitations of region growing

The approximation technique we used, based on GVY region growing, cannot achieve a factor better than three. Our algorithm cuts a cluster C out of the set S, where C is

chosen according to the distance relation x. We allowed ourselves two options for C: the singleton set $\{u\}$ or $B_x(u, 1/2)$. If we restrict ourselves to clusters of the form $B_x(u, r)$, or $\{u\}$, then we are confounded by the following star type example. Admittedly, this example is not an optimal fractional solution to the linear program, but it is a feasible solution and thus Observation 4.1, on which our technique is based, applies.

The positive and negative labels are identical to the previous star, but now every edge has fractional length $1/3$. If our cluster radius is less than $1/3$ then we have a singleton cluster $\{u\}$, in which case the gap ratio is 3. Alternatively, if the radius is at least $1/3$ then all the vertices are in one cluster and the number of mistakes is $n(n-1)/2$. Since the LP cost is $n(n-1)/6 + n/3$, the gap is $3(n-1)/(n+1)$, which tends to 3 as n increases. Therefore, no radius-based approximation algorithm can beat a factor of three.

Using fixed radii

Our factor four algorithm chose between a singleton cluster and a fixed cluster radius of $1/2$. A more general algorithm might select the cluster radius based on the values of the x distance relation. We saw that even if this option were available, we could not achieve an approximation factor better than three. In the remainder of this section, let us restrict our attention to algorithms in which the radius candidates—call them thresholds—for cluster balls are specified *in advance*. We now show that, in some sense, our algorithm ALGCOMPLETE is the best possible we are allowed only one (fixed) threshold.

Theorem 4.3 *Given a set of thresholds, of which k are greater than $1/4$, then our analysis techniques, which rely only on the solution being feasible, cannot be used to show an approximation ratio better than $3 + 1/k$.*

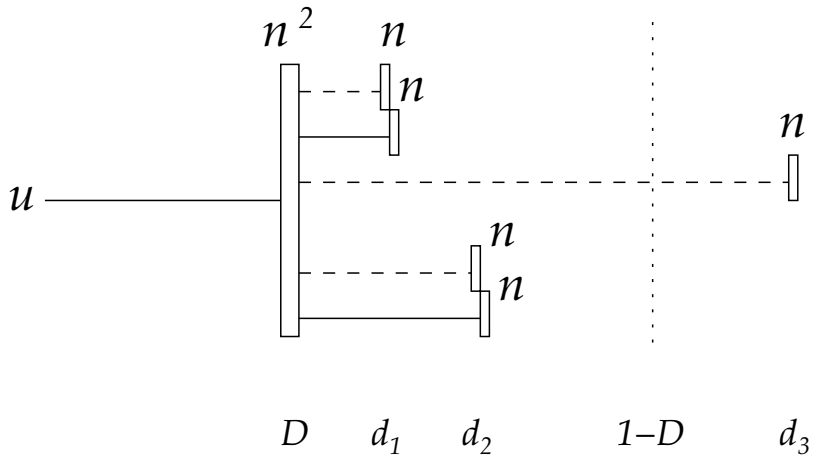

Figure 4.4: Feasible solution example showing that with k thresholds our techniques cannot give an approximation ratio better than $3 + 1/k$. The instance is complete, but we have chosen not to show edges that have little impact on the calculations.

Proof: Consider the analysis of the following feasible solution, shown in Figure 4.4, to the MINDISAGREE LP, which could occur in a single iteration of region growing.

Imagine that there are n^2 vertices at distance $D = k/(3k + 1) - \varepsilon$ from u, and that for each threshold d_i in the range $(D, 1 - D]$ there are n vertices at distance $d_i + \varepsilon$. The edges between the D-vertices and the all of the $d_i + \varepsilon$-vertices are positive. There are also n vertices at distance $d_i - \varepsilon$ for each d_i greater than D (including those thresholds greater than $1 - D$); they have negative edges to the D-vertices. Finally, every edge between u and any other vertex is positive. We ignore all other edges as their costs are dominated by the edges incident to the D-vertices.

For every threshold that lies in the range $(1/4, D)$, the number of mistakes is dominated by n^2 and the LP cost is dominated by Dn^2. Therefore the integrality gap is $1/D$,

which tends to $\rightarrow 3 + 1/k$ as $\varepsilon \rightarrow 0$.

For every other threshold, the LP cost is dominated by the edges between the n^2 D-vertices and the vertices in the other sets. The LP cost of the edges to $d_i - \varepsilon$ and $d_i + \varepsilon$ could be as low as

$$n^3[(d_i + \varepsilon) - D] + n^3[1 - (d_i - \varepsilon) - D] = n^3[1 + 2\varepsilon - 2D]$$
$$\rightarrow n^3 \cdot \frac{k+1}{3k+1} \quad \text{as } \varepsilon \rightarrow 0\,.$$

The LP cost of the negative edges between the D-vertices and the $d_i - \varepsilon$-vertices, where $d_i > D$, could be zero. For each threshold between D and $1-D$, of which there are $k' \leq k$, the number of mistakes is $(k'+1)n^3$. Therefore the ratio of mistakes to LP cost could be as high as

$$\frac{k'+1}{k'} \cdot \frac{3k+1}{k+1}\,,$$

which is $3 + 1/k$ when $k' = k$, and greater otherwise. The total LP cost associated with thresholds whose distance is greater than $1 - D$ may be no greater than before. Since the number of mistakes is *at least* $(k'+1)n^3$, we cannot prove an approximation ratio any better than $3 + 1/k$. ■

Note then that our factor four algorithm, which has one threshold greater than $1/4$, is the best we could hope for with these techniques and just one threshold.

4.4 The connection to feedback edge sets

Using an alternative linear programming formulation, we demonstrate the link between MINDISAGREE in complete graphs and a feedback edge set problem.

$$\begin{aligned} \text{minimize} \quad & \sum_{+(ij)} x_{ij} + \sum_{-(ij)} (1 - x_{ij}) \\ \text{subject to} \quad & \sum_{j=1}^{m-1} x_{i_j,i_{j+1}} - x_{i_m,i_1} \geq 0 \quad \text{for all } C(i_1, \ldots, i_m) \\ & x_{ij} \leq 1 \quad \text{for all } -(ij) \\ & x_{ij} \geq 0 \quad \text{for all } i, j \end{aligned} \tag{4.5}$$

Polygon inequalities are generalizations of triangle inequalities: the length of one edge in a polygon is at most the sum of the lengths of all the other edges in the polygon. A *full* set of polygon inequalities is equivalent to a full set of triangle inequalities. Our new formulation, however, contains only one type of polygon inequality: the length of a negative edge is at most the sum of the lengths of edges in a *positive path* connecting its endpoints. More precisely, for all $i_1, i_2, \ldots, i_m$ such that $+(i_1, i_2), \ldots, +(i_{m-1}, i_m)$, but $-(i_1, i_m)$,

$$\sum_{j=1}^{m-1} x_{i_j,i_{j+1}} - x_{i_1,i_m} \geq 0\,.$$

We call this type of polygon a *negative edge with positive path cycle* (NEPPC), and denote it by $C(i_1, \ldots, i_m)$. Elsewhere [28], NEPPCs have been called erroneous cycles.

We now show that the NEPPC constraints are a sufficiently large set that they imply all the triangle (inequality) constraints for *optimal* solutions to the linear program (4.5). The following simple observation, together with the consequent lemma, is the key.

Observation 4.2 *In an optimal solution to the linear program* (4.5)*, a positive edge either has length zero, or it is part of some tight NEPPC constraint. Likewise, an optimal negative edge either has length one or is part of some tight NEPPC constraint.*

Lemma 4.1 *In an optimal solution to LP* (4.5)*, the polygon inequalities apply to every cycle of positive edges.*

Proof: Consider a positive path p that is incident to both endpoints of positive edge e, with $x_e > x_p$ in an optimal solution (abusing notation). Since the length of e cannot be zero, Observation 4.2 tells us that e lies in some tight NEPPC c. Assume for the moment that c does not share any vertices with p except for the endpoints of e. Now consider the NEPPC c' that is formed by replacing e in c with p. Since c was tight, but p is shorter than e, c' must violate its NEPPC inequality.

It may be that p and c share some vertices other than the endpoints of e. If so, then form a NEPPC c' by building a positive path p' in the following way, where ν refers to the negative edge in c (see also Figure 4.5).

1. Start at one endpoint of ν and walk along c until it intersects p.
2. Now start at the other endpoint of ν and walk in the other direction
   along c until it intersects p.
3. Complete the path p' by walking along the subpath of p that joins
   the intersection points, but does not include e.

Note that the intersection points above are well-defined, as p must meet c at the very least at the endpoints of e. Clearly p' and ν form an NEPPC c', but the length of p' is bounded by the sum of the lengths of $c - e - \nu$ and of p. Since c was tight,

$$x_\nu = x_{c-\nu} = x_{c-e-\nu} + x_e > x_{c-e-\nu} + x_p \geq x_{p'} \,,$$

hence the NEPPC inequality for c' is breached. ■

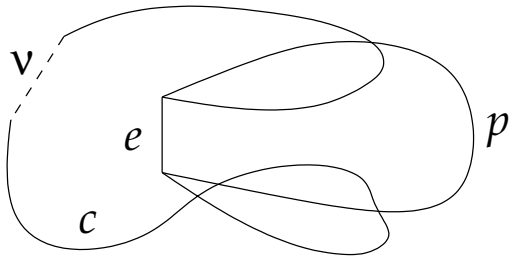

Figure 4.5: Construction of a new NEPPC: Positive edge e is part of a tight NEPPC c, which has one negative edge ν; edge e is also in a cycle with positive path p.

Corollary 4.1 *In every triangle of positive edges the triangle inequalities are satisfied in an optimal solution to* (4.5)*.*

We are now able to prove our main result of this section.

Theorem 4.4 *The linear program with only NEPPC polygon constraints* (4.5) *is equivalent to the triangle inequality program* (4.2)*, in the sense that their sets of optimal solutions are the same.*

Proof: We first show that any optimal solution to (4.5) must satisfy the triangle inequalities.

Although the corollary above deals with all-positive triangles, there are still a number of different cases and configurations to consider. We therefore leave the details to the reader, but note the following general principles of the proof technique.

Consider some triangle in the graph that is not covered by the corollary above: it must have at least one negative edge. If a negative edge has length one, then some of the triangle inequalities are trivially satisfied. Otherwise, the negative edge is contained in a tight NEPPC. The combination of tight NEPPCs and positive triangle edges allows us

$$\begin{aligned} \text{minimize} \quad & \sum_{+(ij)} x_{ij} + \sum_{-(ij)} x'_{ij} \\ \text{subject to} \quad & \sum_{j=1}^{m-1} x_{i_j,i_{j+1}} + x'_{i_m,i_1} \geq 1 \quad \text{for all } C(i_1,\ldots,i_m) \\ & x_{ij} \geq 0 \quad \text{for all } +(ij) \\ & x'_{ij} \geq 0 \quad \text{for all } -(ij) \end{aligned} \tag{4.6}$$

to use either the NEPPC constraints or Lemma 4.1 to be sure that the triangle inequality constraints are observed.

Finally, since the linear program (4.5) is a relaxation of the original (4.2), the two formulations must have the same set of optimal solutions. ■

We note that one can also prove an integral equivalent to Theorem 4.4: any optimal $\{0,1\}$ solution to the NEPPC constraint LP is an optimal solution to the MINDISAGREE problem, in a complete graph.

If we replace each $(1 - x_{ij})$ term with x'_{ij} for each negative edge, we obtain an LP with only positive coefficients (4.6), in which the $x'_{ij} \leq 1$ constraints are unnecessary. In any feasible solution to (4.6), the sum of the terms around any NEPPC is at least 1. If the variables x_{ij} and x'_{ij} are binary, then we have the following interpretation: around any cycle that contains *exactly* one negative edge we must *select* at least one edge. That is, we need a feedback edge set for the set of cycles with exactly one negative edge. If the cycles of interest were those with *at least* one negative edge, we would already have a factor two approximation algorithm [29]. This feedback edge set interpretation might lead to an algorithm with approximation ratio better than four.

As a final comment, we note that there is also some similarity to the notion of *balance*

in *signed* graphs, as used in the social sciences [70]. Each person in some group is represented by a node in a graph; there is an edge between a pair of nodes if there is some strong relationship between the people, with the sign of the edge reflecting the nature of the relationship. A group, and therefore the graph, is called balanced if every cycle in the graph contains an even number of negative edges. There exist linear time algorithms to determine whether a signed graph is balanced. However, some graphs are neither completely balanced nor completely unbalanced and there is ongoing research to measure the degree of balance in them.

Chapter 5

Maximizing agreements

As already mentioned, Bansal, Blum, and Chawla [7] present a PTAS for MAXAGREE on complete graphs, so we focus on general graphs here. Obtaining a $1/2$ approximation for MAXAGREE is trivial, as observed by Bansal *et al.* [7] for the complete graph. If the total weight of positive edges is greater than the total weight of negative edges, place all vertices in one cluster; otherwise, put each of them in an individual cluster.

5.1 A linear program with poor integrality gap

Consider an LP relaxation for MAXAGREE similar to the LP used for MINDISAGREE in Chapter 4. The constraints are exactly the same, but the objective is

$$\text{maximize} \quad \sum_{+(ij)} w_{ij} \cdot (1 - x_{ij}) + \sum_{-(ij)} w_{ij} \cdot x_{ij}$$

Theorem 5.1 *The integrality gap of the LP relaxation for* MAXAGREE *is no better than* $2/3 + \varepsilon$ *for any* $\varepsilon > 0$.

Proof: Our gap instance consists of two sets A and B of n vertices each. The graph is in fact complete, with every edge having a positive or negative label. The edges between A and B are positive; those with end points within the same set are negative. Thus there are n^2 positive edges and $n(n-1)$ negative edges. The optimal LP solution assigns $x_{ij} = 1/2$ for $+(ij)$ and $x_{ij} = 1$ for $-(ij)$, and so $\mathsf{OPT}_{\mathsf{LP}}$ is $n(n-1) + n^2/2$. On the other hand, the value of OPT for this instance is n^2: any instance with equal numbers of elements from A and B in each cluster suffices—we leave the proof to the reader. Hence the integrality gap is $2n/(3n-2)$, which approaches $2/3$ as n increases. ■

5.2 Rounding a semidefinite program

We next consider a semidefinite program (SDP) for MAXAGREE, as SDPs can be solved to arbitrary precision in polynomial time. Lovasz [58] pioneered the application of semidefinite programming, when he used it to determine the value of the ϑ function, an upper bound on the size of the maximum independent set in a graph. Goemans and Williamson [37] were the first to provide an SDP based approximation algorithm with their 0.878-approximation for max cut. This was followed shortly afterwards by Karger, Motwani, and Sudan [50] who used an SDP relaxation to obtain an $O(n^{1-3/(k+1)} \log^{1/2} n)$ coloring for k-colorable graphs.

To motivate the SDP, we associate a distinct basis vector with each cluster in a solution; for every vertex i in that cluster we set the unit vector v_i to be that basis vector. The agreement of the clustering solution can now be expressed in terms of the dot products $v_i \cdot v_j$. If vertices i and j are in the same cluster, then $v_i \cdot v_j = 1$, if not, $v_i \cdot v_j = 0$. With

this vector solution in mind, we consider the SDP relaxation for MAXAGREE (5.1).

$$\begin{array}{lll} \text{maximize} & \displaystyle\sum_{+(ij)} w_{ij}(v_i \cdot v_j) + \sum_{-(ij)} w_{ij}(1 - v_i \cdot v_j) & \\ \text{subject to} & v_i \cdot v_i = 1 & \text{for all } i \\ & v_i \cdot v_j \geq 0 & \text{for all } i, j \end{array} \tag{5.1}$$

Consider the following general approach for rounding this SDP: Pick t random hyperplanes, dividing the set of vertices into 2^t clusters. We refer to this scheme as H_t. Our rounding scheme takes the better of the two solutions returned by H_2 and H_3, denoted by Best(H_2, H_3).

Theorem 5.2 *Best(H_2, H_3) returns a solution in which the expected number of agreements is at least* $0.7664\ \mathsf{OPT}_{\mathsf{SDP}}$.

Proof: In order to analyze Best(H_2, H_3), we consider a slightly different scheme: pick H_2 with probability $1 - \alpha$ and pick H_3 with probability α, denoted by Comb(H_2, H_3). Clearly the approximation ratio of Comb(H_2, H_3) is a lower bound on the approximation ratio of Best(H_2, H_3).

We perform an edge-by-edge analysis: For each edge ij, we measure the expected contribution to the solution produced relative to its SDP contribution. The (nonnegative) edge weights are common to both the integral formulation and its SDP relaxation and so can be ignored. Consider an edge ij such that the angle between v_i and v_j is $\theta \in [0, \pi/2]$. The probability that v_i and v_j are *not* separated by H_t is $(1 - \theta/\pi)^t$.

If ij is a positive edge, the contribution to the SDP solution is $v_i \cdot v_j = \cos\theta$. On the other hand, the expected contribution to the number of agreements in Comb(H_2, H_3) is

$$(1 - \alpha)(1 - \theta/\pi)^2 + \alpha(1 - \theta/\pi)^3.$$

If ij is a negative edge, the contribution to the SDP solution is $1 - v_i \cdot v_j = 1 - \cos\theta$. On the other hand, the expected contribution to the number of agreements in Comb(H_2, H_3) is

$$1 - (1-\alpha)(1-\theta/\pi)^2 - \alpha(1-\theta/\pi)^3.$$

Thus the approximation ratio can be bounded by

$$\min_{\theta\in[0,\pi/2]} \left\{ \frac{(1-\alpha)(1-\frac{\theta}{\pi})^2 + \alpha(1-\frac{\theta}{\pi})^3}{\cos\theta}, \frac{1-(1-\alpha)(1-\frac{\theta}{\pi})^2 - \alpha(1-\frac{\theta}{\pi})^3}{1-\cos\theta} \right\}.$$

For $\alpha \leq 0.1316$, the minimum of the two expressions is $3/4+\alpha/8$. In fact the minimum value of the second expression is $3/4+\alpha/8$ for all $\alpha \in [0,1]$ and is achieved when $\theta = \pi/2$. The upper bound on α is obtained by minimizing the first expression. Setting $\alpha = 0.1316$ yields a 0.7664 approximation. ■

The following simple example shows that the best approximation factor we can hope to achieve using the SDP (5.1) is at most 0.828. Our example has three vertices, $1, 2, 3$, in which edges $(1,2)$ and $(2,3)$ are positive, but $(1,3)$ is negative. The optimal SDP solution consists of the vectors $v_1 = (1,0), v_2 = (1/\sqrt{2}, 1/\sqrt{2}), v_3 = (0,1)$, with objective value $1 + 2/\sqrt{2} = 1 + \sqrt{2}$. On the other hand, $\mathsf{OPT} = 2$, so the integrality gap is at most $2/(1+\sqrt{2}) \approx 0.828$.

Our SDP formulation does not, however, respect the triangle inequalities on the values $x_{ij} = 1 - v_i \cdot v_j$. Even with such constraints added, the example below shows that significant improvements to the approximation ratio may not be possible. Consider an instance on five vertices $0, 1, 2, 3, 4$. Edges from 0 are positive, but all others are negative. With $v_0 = (0.5, 0.5, 0.5, 0.5)$, and v_i equal to the i^{th} basis vector e_i, $\mathsf{OPT}_{\mathsf{SDP}} = 8$. However, $\mathsf{OPT} = 7$, with clusters $\{0,1\}, \{2\}, \{3\}, \{4\}$, showing that we can rule out an

SDP-based algorithm with approximation factor greater than least $7/8$ that observes the triangle inequalities.

An alternative approach is to use the rounding scheme used by Frieze and Jerrum [32] for max-k-cut. The basic idea is to pick k random unit vectors (*spokes*) and assign each vector to the closest spoke. The analysis of such a scheme is quite involved and the gap example above suggests that pursuing this direction is unlikely to yield significant improvements. Nevertheless, Swamy [72] recently carried out an analysis of such a rounding procedure and reported a factor 0.7666 approximation algorithm for MAXAGREE.

5.3 Almost satisfiable instances

Consider an instance for which the optimal SDP solution is $(1-\varepsilon)W$, where W is the total weight of all the edges. We show that in this case it is possible to obtain a clustering with expected agreement in $(1 - O(\sqrt{\varepsilon}\log(1/\varepsilon)))W$. This strong result suggests there would be difficulty in proving super-constant inapproximability for MINDISAGREE.

It is convenient at this point to define various parameters. Let P denote the total weight of the positive edges and N the total weight of the negative edges. We define ρ and ν as follows:

$$\rho = \frac{\sum_{+(ij)} w_{ij}(1 - v_i \cdot v_j)}{P}$$

$$\nu = \frac{\sum_{-(ij)} w_{ij}(v_i \cdot v_j)}{N}.$$

Since $\mathsf{OPT}_{\mathsf{SDP}} = (1-\varepsilon)W$, we observe that $\varepsilon \cdot W = \rho \cdot P + \nu \cdot N$.

Lemma 5.1 $P\sqrt{\rho} \leq W\sqrt{\varepsilon}$.

Proof: It is trivially true if $\rho \leq \varepsilon$. Otherwise, by definition $P\rho \leq W\varepsilon$, so $P\sqrt{\rho} \leq W\varepsilon/\sqrt{\rho} < W\sqrt{\varepsilon}$. ■

We prove that the rounding scheme H_t with $t = \log(1/\varepsilon)$ satisfies the following two lemmas and then conclude with the main result of this section.

Lemma 5.2 *The expected contribution from the positive edges is at least* $P - O(\sqrt{\varepsilon}\log(1/\varepsilon))W$.

Proof: Define ε_{ij} to be $1 - v_i \cdot v_j$, so the expected weight of positive edges that are *not* cut in the solution is

$$\sum_{+(ij)} w_{ij} \left[1 - \cos^{-1}(1 - \varepsilon_{ij})/\pi)\right]^t .$$

The function $(1 - \cos^{-1}(x)/\pi)^t$ is convex, so by applying Jensen's inequality, we obtain the lower bound

$$P\left[1 - \cos^{-1}(1 - \rho)/\pi\right]^t .$$

Since $\cos^{-1}(1 - \rho)$ is in $O(\sqrt{\rho})$, the contribution of the positive edges is at least

$$P(1 - O(\sqrt{\rho}))^t \geq P(1 - tO(\sqrt{\rho})) \geq P - O(\sqrt{\varepsilon}\log(1/\varepsilon))W ,$$

by Lemma 5.1. ■

Lemma 5.3 *The expected contribution from the negative edges is at least* $N(1 - \varepsilon - \nu)$.

Proof: Now redefine ε_{ij} to be $v_i \cdot v_j$. The expected weight of negative edges that *are* cut in the solution is

$$\sum_{-(ij)} w_{ij} \left(1 - \left[1 - \cos^{-1}(\varepsilon_{ij})/\pi\right]^t\right) .$$

Again, convexity tells us that

$$\left[1 - \cos^{-1}(\varepsilon_{ij})/\pi)\right]^t$$

is no greater than

$$\varepsilon_{ij}\left(1 - \cos^{-1}(1)/\pi\right)^t + (1 - \varepsilon_{ij})\left(1 - \cos^{-1}(0)/\pi\right)^t .$$

This is bounded above by $\varepsilon_{ij} + 1/2^t$. Since $N\nu = \sum_{-(ij)} w_{ij}\varepsilon_{ij}$, the expected contribution of the negative edges is at least $N(1 - \nu - \varepsilon)$, for $t = \log(1/\varepsilon)$. ■

Theorem 5.3 *The expected number of agreements as a result of rounding with $H_{\log(1/\varepsilon)}$ is in $W(1 - O(\sqrt{\varepsilon}\log(1/\varepsilon)))$.*

Proof: Lemmas 5.2 and 5.3 show that the expected number of agreements resulting from the $H_{\log(1/\varepsilon)}$ rounding scheme is at least

$$(P + N) - O(\sqrt{\varepsilon}\log(1/\varepsilon))W - (\varepsilon + \nu)N .$$

We note that $(\varepsilon + \nu)N \leq 2\varepsilon W$ and that ε is in $O(\sqrt{\varepsilon}\log(1/\varepsilon))$ as $\varepsilon \to 0$. Therefore the expected number of agreements is at least $W(1 - O(\sqrt{\varepsilon}\log(1/\varepsilon))$. ■

Chapter 6

Maximizing correlation

Until now, no nontrivial approximation algorithm was known for MAXCORR; in this chapter we demonstrate an $\Omega(1/\log n)$ approximation.We start by showing that we can concentrate on the two cluster case. Next we make some remarks about the quadratic program for two clusters and then describe the $\Omega(1/\log n)$ approximation it. We finish with applications for max cut and correlated random variables.

6.1 Reduction to the two cluster case

Let $\mathsf{corr}(\kappa)$ stand for the correlation of clustering κ. There is only one way of placing each item into a singleton cluster: call this clustering κ_n and let $\mathsf{OPT}_n = \mathsf{corr}(\kappa_n)$. In contrast, there are several ways of splitting the items into two clusters, but we let κ_2^{OPT} stand for one of those with maximal correlation OPT_2. Finally, κ^{OPT} is some partitioning that has maximum correlation OPT.

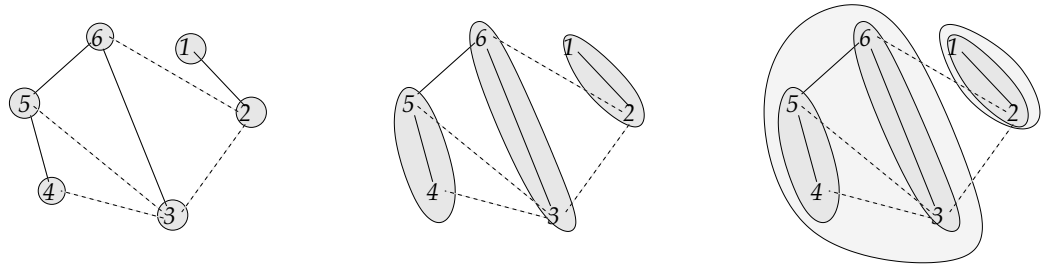

Figure 6.1: Assigning costs in the optimal solution (center) to the singleton (left) and two cluster (right) optimums. The two cluster diagram shows the type of supercluster construction used in the proof.

Lemma 6.1

$$\mathsf{OPT}_n + 2\mathsf{OPT}_2 \geq \mathsf{OPT} .$$

Proof: Let the four quantities, w_+, a_+, w_- and a_-, stand for the numbers of positive (negative) pairs within (across) clusters in our *optimal* solution κ^{OPT}. By definition,

$$\mathsf{OPT} = w_+ - a_+ - w_- + a_- .$$

If we split everything up into singletons, see Figure 6.1 we see that

$$\mathsf{OPT}_n = -w_+ - a_+ + w_- + a_- .$$

Although we cannot calculate OPT_2, we can at least provide a lower bound for it. Consider constructing a partitioning by randomly assigning each *cluster* in κ^{OPT} to one of two new superclusters. The expected correlation of the result of this random procedure is a lower bound for OPT_2. All within-cluster pairs remain within-cluster pairs. With probability $1/2$ each across-cluster pair becomes a within-cluster pair; consequently its

expected contribution to the correlation is zero, and so $\mathsf{OPT}_2 \geq w_+ - w_-$. We now find that

$$\mathsf{OPT}_n + 2\mathsf{OPT}_2 \geq w_+ - a_+ - w_- + a_- = \mathsf{OPT}\,.$$

■

Lemma 6.1 shows that a reasonable approximation to OPT_2 will provide a reasonable approximation algorithm for MAXCORR.

We now adopt a formulation for partitioning into two clusters that is somewhat related to the SDP for MAXAGREE. For each node in the graph we have a ± 1 variable x_i indicating which of the two clusters it belongs to. If $x_i x_j > 0$, then i and j are in the same cluster, but if $x_i x_j < 0$, then they are in different clusters. The contribution from edge ij, with weight w_{ij}, to the correlation is $a_{ij} x_i x_j$, where $a_{ij} = w_{ij}$ for positive edges and $a_{ij} = -w_{ij}$ for negative edges. The obvious quadratic programming formulation to find a κ_2^{OPT} solution is thus

$$\begin{aligned} &\text{maximize} \quad \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} x_i x_j \\ &\text{subject to} \qquad\qquad x_i \in \{-1, 1\} \quad \text{for all } i\,, \end{aligned} \tag{6.1}$$

where $a_{ij} = 0$ if $i \geq j$. It is easy to adjust the a_{ij} terms in the matrix A so that the matrix is symmetric. Let us call the quadratic maximization problem in (6.1) MAXQP, allowing A to be any matrix with null diagonal entries. In the next two sections we explain the background to the quadratic programming problem and then describe an algorithm *ApproxMaxQP* for MAXQP. For the moment, however, let us focus on the algorithm *ApproxMaxCorr* for MAXCORR, whose construction follows directly from the discussion above.

ApproxMaxCorr

1. Construct the matrix A thus:

   if $i < j$ and pair ij is similar then $a_{ij} = w_{ij}$,

   if $i < j$ and pair ij is dissimilar then $a_{ij} = -w_{ij}$;

   otherwise $a_{ij} = 0$.

2. Execute *ApproxMaxQP* on A and obtain solution x.

3. Form partitioning κ_2 by assigning item i to cluster one if $x_i = -1$,

   or to cluster two if $x_i = 1$.

4. Calculate OPT_n and $\mathsf{corr}(\kappa_2)$ and return the clustering with higher correlation.

Lemma 6.2 ApproxMaxCorr *achieves an approximation of* $\alpha/(2+\alpha)$*, where* α *is the approximation factor of* ApproxMaxQP.

Proof: It is easy to verify that the program (6.1) with the variables set in step 1 of the algorithm is a formulation of MAXCORR restricted to just two clusters, both in terms of feasible solutions and objective values. Clearly,

$$\begin{aligned}\max\{\mathsf{OPT}_n, \mathsf{corr}(\kappa_2)\} &\geq t\,\mathsf{OPT}_n + (1-t)\,\mathsf{corr}(\kappa_2) \\ &\geq t\,\mathsf{OPT}_n + (1-t)\,\alpha\,\mathsf{OPT}_2\,,\end{aligned}$$

for all $t \in [0,1]$. If we let $t = \alpha/(2+\alpha)$, then

$$\begin{aligned}\max\{\mathsf{OPT}_n, \mathsf{corr}(\kappa_2)\} &\geq \frac{\alpha}{2+\alpha}\mathsf{OPT}_n + \frac{2\alpha}{2+\alpha}\mathsf{OPT}_2 \\ &\geq \frac{\alpha}{2+\alpha}\mathsf{OPT}\,,\end{aligned}$$

by Lemma 6.1. ■

Theorem 6.2 below shows that the approximation factor of *ApproxMaxQP* is in $\Omega(1/\log n)$. From Lemma 6.2 we therefore conclude:

Theorem 6.1 ApproxMaxCorr *guarantees a solution for* MAXCORR *within* $\Omega(1/\log n)$ *factor of the optimum.*

6.2 Maximizing quadratic programs

Before outlining the approximation algorithm for MAXQP, let us consider this quadratic program in more detail. We are given a matrix A, with null diagonal entries, and asked to maximize $\sum_{i,j} a_{ij}x_ix_j$, subject to $|x_i| = 1$ for all i. We enforce the $a_{ii} = 0$ condition because the terms $a_{ii}x_ix_i$ are equal to a_{ii} and so are just additive constants. Actually it would not harm any of our arguments if we were to allow the a_{ii} values to be nonnegative, but the exposition is simpler if we just ignore these terms. It is important to note that, for $i \neq j$, the a_{ij} terms are arbitrary real values and are not restricted to being nonnegative. The goal is to return a solution that is at least some fraction α of the optimum of (6.1), $\mathsf{OPT}_{\mathsf{QP}}$. Our approximation problem is well defined, for $\mathsf{OPT}_{\mathsf{QP}}$ is strictly positive unless A is the zero matrix (see Lemma 6.6).

We discovered only recently, after we had submitted our research for review [16], that in fact approximation algorithms and integrality gap results similar to ours had already existed for the MAXQP problem [61, 65, 64]. These $\Omega(1/\log n)$ results were not well known in the theoretical computer science community. Our approximation algorithm in Section 6.3 is an algorithmic version of Megretski's integrality gap proof [61].

Prior to our exploration of MAXCORR, Alon and Naor [2] announced approxima-

tion algorithms for a problem similar to MAXQP. Specifically, they approximated the cut norm by trying to maximize

$$\sum_{i=1}^{m}\sum_{j=1}^{n} a_{ij}x_i y_j, \quad \text{s.t. } x_i, y_j \in \{-1, 1\} \text{ for all } i, j\,. \tag{6.2}$$

We can cast (6.2) as an instance of (6.1) by letting

$$z = \begin{pmatrix} x \\ \\ y \end{pmatrix} \quad \text{and} \quad \hat{A} = \begin{bmatrix} 0 & A \\ \\ 0 & 0 \end{bmatrix}, \tag{6.3}$$

so (6.2) is now $z^T\hat{A}z$. Seeing that MAXQP is an extension of Alon and Naor's problem, we spend some time detailing the similarities and differences between them.

A quadratic program may be used to model a graph optimization problem, such as MAXCORR, in which a_{ij} represents some property of the edge between i and j. One immediate observation from (6.3) is that if (6.2) represents the maximization of a function on a *bipartite* graph, then (6.1) represents the maximization of the same function on a *complete* graph.

Alon and Naor's problem has two key properties that ours does not. Firstly, different rounding techniques can be used for the $\{x_i\}$ and the $\{y_j\}$ in (6.2). Secondly, the minimum and maximum values of (6.2) are equal in magnitude (simply replace y with $-y$ in one of the extreme solutions). It turns out that these facts are crucial to their analysis.

Both (6.1) and (6.2) have canonical semidefinite relaxations, respectively,

$$\begin{aligned} \text{maximize} \quad & \sum_{i,j} a_{ij}\, v_i \cdot v_j \\ \text{subject to} \quad & v_i \cdot v_i = 1 \qquad \text{for all } i \\ & v_i \in \mathbb{R}^n \qquad \text{for all } i\,, \end{aligned} \tag{6.4}$$

and

$$\begin{aligned} \text{maximize} \quad & \sum_{i,j} a_{ij}\, u_i \cdot v_j \\ \text{subject to} \quad & u_i \cdot u_i = v_j \cdot v_j = 1 \qquad \text{for all } i, j \\ & u_i, v_j \in \mathbb{R}^n \qquad \text{for all } i, j\,. \end{aligned} \tag{6.5}$$

Unfortunately the term-by-term analysis that Goemans and Williamson [37] used in their 0.878 SDP-based approximation algorithm for max cut—which is, as we explore in Section 6.4, a shifted special case of MAXQP—fails for both (6.4) and (6.5) because some of the terms might be negative. Nevertheless, Grothendieck's inequality [40], a key result in functional analysis, states that the integrality gap between (6.5) and (6.2) is in fact a constant. The exact value of this constant is not known, but Alon and Naor converted proofs of the existence of the constant bound [69, 57] into approximation algorithms for (6.2).

Alon and Naor's first algorithm is deterministic: an explicit set of fourwise independent vectors in $\{-1, 1\}^n$ is constructed. They showed that there exists one such vector whose projections onto the optimal SDP solution vectors, truncated to lie in $[-1, 1]$, give a fractional solution no less than $1/27$ of optimum (the beginning of Section 6.3 shows why a good fractional solution is sufficient). The second algorithm is simply the random hyperplane split of Goemans and Williamson, but with a more involved analysis. This technique treats the x and y variables identically, but the analysis cannot be applied to our problem because it relies on the ratio of the absolute values of the maximum and minimum of the quadratic program being at least some constant. The third algorithm shows that there exist two new families of vectors, $\{u'_i\}$ and $\{v'_j\}$, so that the expected value of every $x_i y_j$ term obtained by splitting the u' and v' vectors with a random hyperplane is some common constant multiple of the corresponding $u_i \cdot v_j$ value. These u' and

v' vectors are found by maximizing a semidefinite program. None of these techniques seems to apply directly to MAXQP, though we adapt some of the ideas.

6.3 Approximating MAXCORR: the two cluster case

We start by showing that the optimum value of MAXQP, modified so that the variables are allowed to take any value in the range $[-1, 1]$, is no larger than that of the original problem (6.1). Consider the following randomized rounding technique for some fractional solution $y \in [-1, 1]^n$:

$$x_i = \begin{cases} -1, \text{ with probability } \frac{1-y_i}{2} \\ \\ +1, \text{ with probability } \frac{1+y_i}{2} \end{cases}, \tag{6.6}$$

where each x_i is rounded independently of the others. A simple calculation shows that, for $i \neq j$, $\mathbf{E}[x_i x_j] = y_i y_j$. Since the a_{ii} terms are zero, the expected objective value of the *integral* solution equals that of the fractional solution.

So, to the approximation algorithm *ApproxMaxQP*. We first solve the semidefinite relaxation (6.4) of the quadratic program in polynomial time, up to arbitrary precision. We then round the SDP *vector* solution $\{v_i\}$ to a fractional solution y. It may be that some y_i values fall outside the range $[-1, 1]$; if so, we truncate them to ± 1. We show that this happens so rarely that the truncation does not alter the expected value of the solution significantly. Finally, we use the rounding technique (6.6) in the previous paragraph to obtain a $\{-1, 1\}$ solution x.

The solution x is clearly in $\{-1, 1\}^n$, and it was obtained in polynomial time. A sequence of lemmas in the remainder of this section will build up to the following result.

ApproxMaxQP

1. Obtain an optimal solution $\{v_i\}$ to the SDP (6.4).

2. Create vector r in which the r_i are drawn independently from the unit Normal distribution.

3. Let $z_i = v_i \cdot r/T$, where $T > 0$ will be specified later.

4. If $|z_i| > 1$, then $y_i = z_i/|z_i|$, otherwise $y_i = z_i$.

5. Obtain x_i from y_i using rounding procedure (6.6).

Theorem 6.2 *The ratio of the expected value of the solution x returned by* ApproxMaxQP *to the maximum value of the quadratic program* (6.1) *is in $\Omega(1/\log n)$, if $T = \sqrt{4 \log n}$.*

Lemma 6.3 *The expected value of $z_i z_j$ is $v_i \cdot v_j/T^2$.*

Proof: Since the distribution of the r vector is *spherically symmetric*, we can assume that $v_i = e_1$ and $v_j = ae_1 + be_2$. Therefore $Tz_i = v_i \cdot r = r_1$ and $Tz_j = v_j \cdot r = ar_1 + br_2$. Hence

$$\begin{aligned} T^2\mathbf{E}[z_i z_j] &= a\mathbf{E}[r_1^2] + b\mathbf{E}[r_1 r_2] \\ &= a\mathbf{Var}[r_1] + b\mathbf{E}[r_1]\mathbf{E}[r_2] \\ &= a = v_i \cdot v_j \,. \end{aligned}$$

■

If we were lucky and every $|z_i|$ were at most 1, then we would have a $1/T^2$ approximation, since the optimum value of (6.4) is at least the optimum of (6.1). Since this might not happen, we need to analyze the truncated solution y. We show that the expected value of $\Delta_{ij} = z_i z_j - y_i y_j$ is small in magnitude.

Lemma 6.4 $|\mathbf{E}[\Delta_{ij}]|$ *is less than* $8e^{-T^2/2}$.

Proof: Let us consider the expected value of $|\Delta_{ij}|$ on various regions (of r). We assume that n is sufficiently large that $T \geq 1$. On the region $S = \{r : |z_i| \leq 1, |z_j| \leq 1\}$, $y_i = z_i$ and $y_j = z_j$, so $\mathbf{E}_S[\Delta_{ij}] = 0$.

Now, due to rotational symmetry, we may again assume that $v_i = e_1$ and $v_j = ae_1 + be_2$. So the probability that r lies in the region $B = \{r : z_i > 1\}$ is $\mathbf{Pr}[r_1 > T] = 1 - \Phi(T)$, where Φ is the cdf for the Normal distribution. Therefore,

$$\mathbf{E}_B[|y_i y_j|] \leq \mathbf{E}_B[1] = \mathbf{Pr}[r \in B] = 1 - \Phi(T)\,. \tag{6.7}$$

Furthermore,

$$\mathbf{E}_B[T^2|z_i z_j|] = \int_{-\infty}^{+\infty} \int_{T}^{+\infty} |s(as + bt)| \frac{1}{2\pi} e^{-s^2/2} e^{-t^2/2} ds\, dt\,. \tag{6.8}$$

Let us consider each term of (6.8) one at a time,

$$\begin{aligned} \int_T^\infty s^2 e^{-s^2/2} ds &= -se^{-s^2/2}\Big|_T^\infty + \int_T^\infty e^{-s^2/2} ds \\ &= Te^{-T^2/2} + \sqrt{2\pi}(1 - \Phi(T))\,. \end{aligned}$$

Also,

$$\int_T^\infty |s| e^{-s^2/2} ds = e^{-T^2/2} \quad \text{and} \quad \int_{-\infty}^{+\infty} |t| e^{-t^2/2} dt = 2\,.$$

Putting it all together, we see that

$$\begin{aligned} \mathbf{E}_B[T^2|z_i z_j|] &\leq \left(\frac{|a|T}{\sqrt{2\pi}} + \frac{|b|}{\pi}\right) e^{-T^2/2} + |a|(1 - \Phi(T)) \\ &< Te^{-T^2/2} + (1 - \Phi(T)), \end{aligned} \tag{6.9}$$

as $|a|, |b| \leq 1$. Since $T \geq 1$, combining (6.7) and (6.9), we have

$$\mathbf{E}_B[|\Delta_{ij}|] \leq \mathbf{E}_B[|z_i z_j| + |y_i y_j|] < \frac{e^{-T^2/2}}{T} + 2(1 - \Phi(T))\,,$$

where the first inequality is merely the triangle inequality.

By symmetry, we will have the same result on the region $\{r : z_i < -1\}$. As there was nothing special about i, the same bound also applies for the regions $\{r : z_j > 1\}$ and $\{r : z_j < -1\}$. The union of these four regions is the complement of the set S. Since the function $|\Delta_{ij}|$ is nonnegative, its expectation on $\overline{S}$ is less than

$$\frac{4}{T}e^{-T^2/2} + 8(1 - \Phi(T)) \,. \tag{6.10}$$

But $\mathbf{E}_S[\Delta_{ij}]$ is 0, so (6.10) is a bound on $\mathbf{E}[|\Delta_{ij}|]$, and hence on $|\mathbf{E}[\Delta_{ij}]|$. We can bound the second term of (6.10) by

$$4\int_T^\infty te^{-t^2/2}dt = 4e^{-T^2/2} \,,$$

as $T \geq 1$. Therefore, $|\mathbf{E}[\Delta_{ij}]| < 8e^{-T^2/2}$. ■

We are now close to obtaining an $\Omega(1/\log n)$ approximation for MAXQP. Let $\mathsf{OPT}_{\mathsf{SDP}}$ stand for the optimum value of (6.4).

Lemma 6.5

$$\mathbf{E}[\sum_{i,j} a_{ij}y_iy_j] > \frac{\mathsf{OPT}_{\mathsf{SDP}}}{T^2} - 8e^{-T^2/2}\sum_{i,j}|a_{ij}|$$

Proof:

$$\mathbf{E}[\sum_{i,j} a_{ij}y_iy_j] = \mathbf{E}[\sum_{i,j} a_{ij}z_iz_j] + \mathbf{E}[\sum_{i,j} a_{ij}(-\Delta_{ij})]$$

From Lemma 6.3, we know that the first term of the right hand side is $\mathsf{OPT}_{\mathsf{SDP}}/T^2$. The

second term is

$$\begin{aligned} -\mathbf{E}[\sum_{i,j} a_{ij}\Delta_{ij}] &= -\sum_{i,j} a_{ij}\mathbf{E}[\Delta_{ij}] \\ &\geq -\left|\sum_{i,j} a_{ij}\mathbf{E}[\Delta_{ij}]\right| \\ &\geq -\sum_{i,j} |a_{ij}||\mathbf{E}[\Delta_{ij}]|\,, \end{aligned}$$

which Lemma 6.4 proves is greater than $-8e^{-T^2/2}\sum_{i,j}|a_{ij}|$. ■

The obvious next step is to show that this *error* term is insignificant. Recall that $\mathsf{OPT}_{\mathsf{QP}}$ stands for the optimum value of (6.1).

Lemma 6.6

$$\mathsf{OPT}_{\mathsf{QP}} \geq \frac{1}{n}\cdot\sum_{i,j}|a_{ij}|$$

Proof: Consider constructing a random matching on an n-vertex complete graph in the following way. Select an edge uniformly at random, remove the endpoints from the graph, and repeat. It is easy to show that the probability of an edge being included in the matching is $1/(n-1)$ if n is even and $1/n$ if n is odd. Now, if we assign to each edge the weight $|a_{ij}|$, then there exists some matching on the n vertices of total weight at least $\sum_{i,j}|a_{ij}|/n$ (the expected value under this random construction).

Given a matching, we randomly construct a vector $x \in \{-1,1\}^n$ whose expected QP objective value is the same as the matching weight. For each edge ij in the matching, we independently set x_i to ± 1 uniformly at random; we let x_j equal x_i if and only if a_{ij} is nonnegative. The unmatched vertex, if n is odd, is assigned a value uniformly and

independently. For each matched pair ij we score $|a_{ij}|$, for every other pair the expected score is 0, hence the same total as the matching. Therefore $\mathsf{OPT}_{\mathsf{QP}}$ is at least $\sum_{i,j} |a_{ij}|/n$.

■

We can now prove Theorem 6.2.

Proof: Substituting Lemma 6.6 into the statement of Lemma 6.5, with $T = \sqrt{4 \log n}$, we see that

$$\mathbf{E}[\sum_{i,j} a_{ij} y_i y_j] > \mathsf{OPT}_{\mathsf{QP}}[\frac{1}{4 \log n} - 8n^{-2} \cdot n] \,,$$

which is in $\Omega(\mathsf{OPT}_{\mathsf{QP}} / \log n)$. Finally, note that

$$\mathbf{E}[\sum_{i,j} a_{ij} x_i x_j] = \mathbf{E}[\mathbf{E}[\sum_{i,j} a_{ij} x_i x_j \mid y]] = \mathbf{E}[\sum_{i,j} a_{ij} y_i y_j].$$

■

6.4 The relationship with max cut

A random assignment of items in a max cut instance will on average result in a cut of $1/2$ of the total weight of edges. We say that a cut has *gain* δ if a $1/2 + \delta$ fraction of the total weight of edges lie across the cut. The gain is analogous to the idea of the advantage over a random assignment [44].

Maximizing the gain in the max cut problem can easily be formulated as a quadratic program. Let w stand for the total weight of edges in a max cut instance and δ^* stand for the gain of the optimal cut—the optimal cut is of size $w(1/2 + \delta^*)$. The Goemans-Williamson algorithm guarantees only that its solution will have a cut of at least $w(0.439 + 0.878\delta^*)$, which may have no gain at all. Zwick's outward rotations method [80] or Feige

and Langberg's RPR^2 algorithm for LIGHTMAXCUT [30] might approximate the gain well. In fact, our algorithm is an instantiation of Feige and Langberg's rounding scheme, using an s-linear function. These authors do not, however, present any theoretical analysis that applies to our setting.

We can express max cut as a special case of quadratic programming by setting A as follows. For all pairs $i < j$ for which there exists an edge of weight w_{ij}, let $a_{ij} = -w_{ij}$; for all other values of i and j, let $a_{ij} = 0$. For a given solution $x \in \{-1, 1\}^n$, the value of the quadratic program $q(x)$, the value of the cut $k(x)$ and the value of its gain $g(x)$, satisfy $q(x) = 2k(x) - w = 2w\, g(x)$. With this formulation, we can now provide an approximation algorithm for the gain of max cut; the related hardness of approximation result for MAXQP is deferred to Chapter 7.

Theorem 6.3 *If δ^* is the optimum gain of a max cut instance,* ApproxMaxQP *returns a solution whose gain is in*

$$\Omega\left(\frac{\delta^*}{\log(1/\delta^*)}\right).$$

Proof: Lemma 6.5 says that *ApproxMaxQP* will return a solution x for which $q(x)$ is at least

$$\frac{\mathsf{OPT}_{\mathsf{QP}}}{T^2} - 8e^{-T^2/2}w\,, \tag{6.11}$$

as w is the sum of the $|a_{ij}|$ terms. By definition, $\mathsf{OPT}_{\mathsf{QP}} = 2w\delta^*$, so if we set $T = \sqrt{32\log(1/\delta^*)}$ then it is not hard to show that (6.11) is greater than

$$\frac{\mathsf{OPT}_{\mathsf{QP}}}{64\log(1/\delta^*)}, \quad \text{for all } \delta^* \le 1/2\,.$$

Since the gain, $g(x)$, is a positive constant multiple of $q(x)$, the result follows. ∎

6.5 Correlated random variables and distributions on cuts

We make some general observations about rounding solutions to the MAXQP SDP relaxation (6.4) and point out some interesting connections to generating $\{-1, 1\}$ random variables with given correlations. The value of the SDP solution is $\sum_{i,j} a_{ij}\, v_i \cdot v_j$. Suppose we could generate $\{-1, 1\}$ random variables X_i such that $\mathbf{E}[X_i X_j] = C v_i \cdot v_j$ for all i, j and some constant C; this would immediately lead to a C-approximation algorithm. In fact, Alon and Naor's third algorithm, based on Krivine's proof, is exactly of this form: they transform the vectors u_i, v_j into new vectors u_i', v_j', apply random hyperplane rounding to these new vectors, and obtain a distribution on $\{-1, 1\}$ random variables with the desired property. We reiterate that, in doing this, they crucially use the fact that they can apply one transformation for the $\{u_i\}$ and a another one for the $\{v_j\}$; hence this technique does not apply to rounding for MAXQP.

One might wonder whether the existence of an appropriate distribution on $\{-1, 1\}$ random variables is a lucky coincidence. We now demonstrate (the possibly surprising fact) that such a distribution always exists.

In order to do this, we write an LP formulation for the maximum gap of the SDP for a *fixed* vector solution. Given a set of vectors v_i produced by an optimal SDP solution, consider the problem of finding the A matrix that maximizes the gap between $\mathsf{OPT}_{\mathsf{SDP}}$

and $\mathsf{OPT}_{\mathsf{QP}}$. We can represent this with the following LP, in which the a_{ij} are variables

$$\begin{aligned} &\text{minimize} && c \\ &\text{subject to} && \sum_{ij} a_{ij}\, v_i \cdot v_j = 1 \\ &&& \sum_{ij} a_{ij}\, x_i x_j \le c \quad \text{for all } x \in \{-1,1\}^n \end{aligned} \tag{6.12}$$

The accompanying dual program (6.13) has one p_x variable for every possible setting of $x \in \{-1,1\}^n$.

$$\begin{aligned} &\text{maximize} && b \\ &\text{subject to} && \sum_{x} p_x = 1 \\ &&& \sum_{x} p_x\, x_i x_j = b\, v_i \cdot v_j \quad \text{for all } i,j \end{aligned} \tag{6.13}$$

These p_x values specify a probability distribution on the x_is. We know that the primal LP (6.12) has optimal value no lower than the gap of the SDP for MAXQP. By duality, there exists a distribution on $\{-1,1\}$ random variables such that $\mathbf{E}[x_i x_j] = b\, v_i \cdot v_j$, where b, which could be a function of n, is at least the worst case gap of the SDP for MAXQP.

Our analysis in Section 6.3 shows that the gap of the MAXQP SDP is in $\Omega(1/\log n)$, which has the following interesting interpretation. Given a positive semidefinite matrix K, it is well known that there exist correlated Normal random variables X_i such that $\mathbf{E}[X_i X_j] = k_{ij}$. What if you wanted $\{-1,1\}$ random variables instead? Our results show that if we scale the (off-diagonal) entries of the correlation matrix by some factor in $O(1/\log n)$, then we can guarantee the existence of $\{-1,1\}$ random variables with the appropriate correlations.

Chapter 7

Hardness of approximation

In this chapter we present some hardness of approximation results to accompany the previous approximation algorithms. These are summarized at the lower parts of Table 3.1. The proofs in this chapter are essentially in increasing order of intricacy.

7.1 MINDISAGREE in general graphs

We first show that minimum multicut reduces in an approximation preserving way to MINDISAGREE. Note that Bansal *et al.* [7] make a similar observation, though they use the all-pairs version of multicut, usually called multiway cut, for the reduction. Reducing from the more general multicut problem, as other groups have also done independently [25, 28], provides us with evidence of the difficulty of approximating MINDISAGREE within any constant factor. In contrast, multiway cut has approximation algorithms with performance ratio a very small constant, 1.3438 being the current best [13, 49].

Theorem 7.1 *Minimum multicut reduces in an approximation preserving way to* MINDISAGREE.

Proof: Given a graph G with k pairs (s_i, t_i), in which each s_i must be separated from each t_i, form an instance H of MINDISAGREE. The edges of G become positive edges in H with unit weight. For each i, $1 \leq i \leq k$, we add a (negative) edge between s_i and t_i with weight $-W$ for some large positive integer W, say $W = n^2$. We can make the instance unweighted by replacing a negative edge of weight $-W$ by W parallel length two paths; each path has a fresh intermediate vertex, with one edge of weight 1 and the other of weight -1. Clearly, the minimum cost clustering must have s_i and t_i in different clusters for every i. The cost of the solution is simply the number of positive edges that lie between clusters, which is the same as the cost of the multicut. ■

Since minimum multicut is known to be APX-hard [36], we conclude that MINDISAGREE is also APX-hard. Furthermore, an improvement over the $O(\log n)$ approximation ratio, which we matched in Section 4.1, would solve one of the major open problems in the area of approximation algorithms: Can minimum multicut be approximately solved within a factor in $o(\log n)$?

We also note the following fact concerning the perceived difficulty of multicut which does not seem to have been explicitly pointed out in the literature. It is well known that minimum edge deletion graph bipartization (also known as min uncut) reduces to minimum multicut in an approximation preserving way. The factor $O(\log n)$ approximation for min uncut works by reducing it to a multicut instance on which the GVY algorithm is run [35]. It is implicit in Khot's work [53] that a certain conjecture about Unique games

would result in min uncut being NP-hard to approximate within any constant factor. Therefore, under the same conjecture, it is NP-hard to approximate minimum multicut, and therefore also MINDISAGREE, within any constant factor.

Emanuel and Fiat [28] also present an approximation preserving reduction in the reverse direction to Theorem 7.1, from MINDISAGREE to minimum multicut. This shows that the approximability of MINDISAGREE is identical to that of the fundamental minimum multicut problem.

In the next section, we study the maximization version. As a corollary of our hardness result for MAXAGREE, we will also record an explicit constant factor hardness for MINDISAGREE (Theorem 7.3).

7.2 MAXAGREE in general graphs

Bansal *et al.* [7] provided evidence for the APX-hardness of MAXAGREE by showing that a PTAS for MAXAGREE would lead to a polynomial time algorithm for $O(n^{\varepsilon})$ coloring a 3-colorable graph for every $\varepsilon > 0$. However, the issue of a concrete NP-hardness result for approximating MAXAGREE remained open and is resolved here.

Theorem 7.2 *For every $\varepsilon > 0$, it is NP-hard to approximate the* weighted *version of* MAXAGREE *within a factor of $79/80 + \varepsilon$. Furthermore, it is NP-hard to approximate the* unweighted *version of* MAXAGREE *within a factor of $115/116 + \varepsilon$.*

Proof: We reduce from MAX 3SAT, which is NP-hard to approximate within a factor of $7/8 + \varepsilon$, even on satisfiable instances [43]. Let ϕ be an instance of MAX 3SAT with variables $x_1, x_2, \ldots, x_n$ and clauses $C_1, C_2, \ldots, C_m$. We also assume that for each i, x_i

and $\bar{x}_i$ each appear in the same number of clauses; this is a minor restriction and the inapproximability result for MAX 3SAT stands.

Construct a graph G with integer edge weights from the instance ϕ as follows. The vertices of G are a *root* vertex r, *variable* vertices $x_i, \bar{x}_i$ for $1 \leq i \leq n$, and *clause* vertices c_{1j}, c_{2j}, c_{3j} for each clause C_j, $1 \leq j \leq m$. The edges and their weights are defined as follows (see also Figure 7.1):

- The root r is connected to each c_{pj}, $p = 1, 2, 3$, by a weight 1 edge, and is connected to x_i and $\bar{x}_i$ by a weight B_i edge, where B_i is the number of clauses in which x_i (and $\bar{x}_i$) appears.
- A weight $-B_i$ edge connects x_i and $\bar{x}_i$ for each $i = 1, 2, \ldots, n$.
- The vertices c_{1j}, c_{2j}, c_{3j} corresponding to each clause form a triangle with weight -1 edges.
- Finally, if the p^{th} variable in clause C_j is x_i, for $p = 1, 2, 3$ (assuming some fixed ordering of variables in each clause), then a weight -1 edge connects c_{pj} with x_i.

We now prove that the optimum value of G as an instance of MAXAGREE is $9m + \mathsf{OPT}_\phi$, where OPT_ϕ is the maximum number of clauses of ϕ that can be simultaneously satisfied.

To that end, we show that any clustering can be modified to a specific format, still maximizing the number of agreements. Since the only positive edges incident to x_i and $\bar{x}_i$ are the edges joining them to r, each of x_i and $\bar{x}_i$ can be assumed to be either a singleton cluster or part of the cluster containing r. If both x_i and $\bar{x}_i$ are in the cluster with r, then

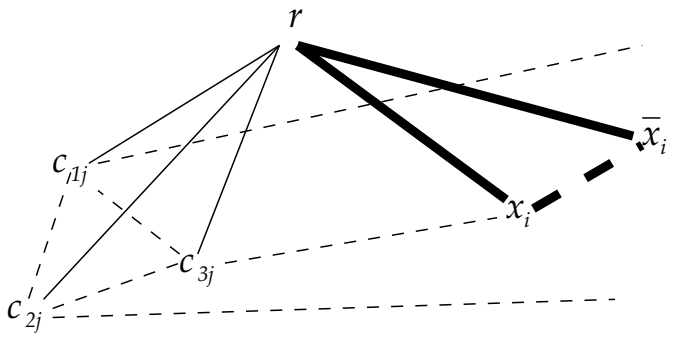

Figure 7.1: Reduction from a MAX 3SAT to a MAXAGREE instance. The jth clause has three vertices c_{1j}, c_{2j}, c_{3j}. The ith variable has two vertices $x_i, \bar{x}_i$. Solid lines represent positive edges, dashed negative edges; thick lines represent edges of weight B_i.

we can make one of them, say x_i, a singleton and the number of agreements will not decrease, since we will lose B_i for the edge (r, x_i), but will gain B_i for the edge $(x_i, \bar{x}_i)$. Similarly, if both x_i and $\bar{x}_i$ are singletons, we can place x_i in the cluster containing r — we will gain a value of B_i for the edge (r, x_i) and might lose at most a value of B_i for the edges connecting x_i to the appropriate c_{pj}s.

Once in this format, a clustering corresponds to a truth assignment to the variables of ϕ in a natural way: variable x_i is true if it is in a singleton cluster, but false if it is in the root-cluster. Now for each clause C_j, we can cluster the vertices c_{pj}, $p = 1, 2, 3$, in the following way without decreasing the number of agreements. If C_j is not satisfied by the above assignment, which means all its literals are in the r-cluster, we place each c_{pj} in a singleton cluster for $p = 1, 2, 3$. If C_j is satisfied, say because its first literal is set true, then we place c_{1j} in the r-cluster, but c_{2j} and c_{3j} in singleton clusters. Consequently, we have four agreements: the negative edges between the c_{pj}s and the positive edge (c_{1j}, r).

The negative weight edges between c_{1j}, c_{2j}, and c_{3j} ensure that, regardless of how many of C_j's literals are true, we always achieve the same number of agreements whenever C_j is satisfied.

It is easily seen that the total weight of correctly clustered edges equals

$$\Big(\sum_{i=1}^{n} 2B_i\Big) + 6m + m^* = 9m + m^*\,,$$

where m^* is the number of clauses satisfied by the above assignment. Therefore the optimum value of this instance of MAXAGREE is $9m+\mathsf{OPT}_\phi$. The claimed result follows since distinguishing between the cases $\mathsf{OPT}_\phi = m$ and $\mathsf{OPT}_\phi \leq (7/8+\varepsilon)m$ is NP-hard [43].

In order to obtain a result for unweighted (± 1)-labeled graphs, we replace each positive (resp. negative) edge of weight B_i (resp. $-B_i$) by B_i length-two paths whose edges have weights $1, 1$ (resp. $1, -1$), as in the proof of Theorem 7.1. Now, if a weight B_i (positive or negative) edge is correctly clustered, then all the $2B_i$ newly constructed edges agree with the labeling; otherwise we get only B_i agreements. Using this gadget, we conclude that there is a $115/116+\varepsilon$ inapproximability factor for the unweighted version of MAXAGREE; we omit the straightforward calculations. ■

Since the number of disagreements in an optimum clustering is simply the sum of the weights of edges minus the number of agreements, the above reduction also establishes the following two theorems.

Theorem 7.3 *For every $\varepsilon > 0$, it is NP-hard to approximate both the* weighted *and* unweighted *versions* MINDISAGREE *within a factor of $29/28-\varepsilon$.*

Theorem 7.4 *For every $\varepsilon > 0$, it is NP-hard to approximate the* weighted *version of* MAXCORR *within a factor of $25/26+\varepsilon$, and the* unweighted *version within a factor of $43/44+\varepsilon$.*

At this point we digress to consider the difficulty of approximating the related problem MAXQP. The reduction we used to obtain an approximation algorithm for maximizing the gain of a cut also leads to a hardness result for MAXQP.

Lemma 7.1 *For all $\varepsilon > 0$, it is NP-hard to approximate* MAXQP *within factor* $11/13 + \varepsilon$.

Proof: Let α stand for the hardness factor for max cut. Since $\mathsf{OPT}_{\mathsf{QP}} = 2\mathsf{OPT}_{\mathsf{CUT}} - w$, distinguishing between the cases $\mathsf{OPT}_{\mathsf{CUT}} \geq k$ and $\mathsf{OPT}_{\mathsf{CUT}} < \alpha k$ is equivalent to distinguishing between $\mathsf{OPT}_{\mathsf{QP}} \geq 2k - w$ and $\mathsf{OPT}_{\mathsf{QP}} < 2\alpha k - w$. The ratio of these two bounds on $\mathsf{OPT}_{\mathsf{QP}}$ is

$$\frac{2\alpha k - w}{2k - w}.$$

The lemma follows from the $16/17 + \varepsilon$ hardness result for max cut [43, 73], which holds for $k = 17w/21$. ■

7.3 MINDISAGREE in complete graphs

In addition to their constant factor approximation algorithm, Bansal *et al.* [7] proved the NP-completeness of MINDISAGREE on *complete* graphs. Their reduction does not yield any hardness of approximation result, but they do show that the maximization version admits a PTAS on complete graphs. Theorem 7.5, nicely completes the picture of the complexity of the problem on complete graphs, complementing our factor four approximation algorithm.

Theorem 7.5 *There exists some constant $c > 1$ for which it is NP-hard to approximate* MINDISAGREE *on* complete graphs *within a factor of c.*

Proof: We give a reduction from the max 2-colorable subgraph problem on bounded degree 3-uniform hypergraphs. Here the input is a 3-uniform hypergraph $H = (V, S)$ where each hyperedge in $S = \{e_1, e_2, \ldots, e_m\}$ consists of three elements of $V = \{v_1, \ldots, v_n\}$ with the added restriction that each element of V occurs in at most B hyperedges, for some absolute constant B (so that $m \leq Bn/3$). The goal is to find a 2-coloring of V that maximizes the number of hyperedges that are *split* by the coloring, that is, are bichromatic. It is known that for some absolute constants $\gamma > 0$ and B (integer), given such a 3-uniform hypergraph it is NP-hard to distinguish between the following two cases: (i) H is 2-colorable, i.e., there exists a 2-coloring of its vertices under which no hyperedge is monochromatic, and (ii) every 2-coloring of V leaves at least a fraction γ of hyperedges in S monochromatic. This follows for example from the reduction used to show the hardness of max 3-set splitting [41]. The starting point for that reduction is a constraint satisfaction problem called MAXSNE_4 [41], that Håstad has shown to be hard to approximate [43]. This hardness result also holds under a bounded occurrence restriction; therefore the 3-uniform hypergraph constructed by the reduction from MAXSNE_4 can also be assumed to have degree bounded by an absolute constant B.

The first step in the reduction is to construct a graph G from the hypergraph H. This step is analogous to the reduction from MAX 3SAT to 3-dimensional matching in Section 9.4 of Papadimitriou's text [67] and is sketched in Figure 7.2. Specifically, for each $v_i \in V$, we construct a *flower* structure F_i with $4s_i$ vertices U_i, where $s_i \leq B$ is the number of hyperedges in which v_i occurs. The set U_i consists of $2s_i$ vertices that form an induced cycle, together with $2s_i$ *petal* vertices each of which is adjacent to the two endpoints of one of the $2s_i$ cycle edges. Let O_i (resp. E_i) be the petal vertices with odd (resp. even) indices

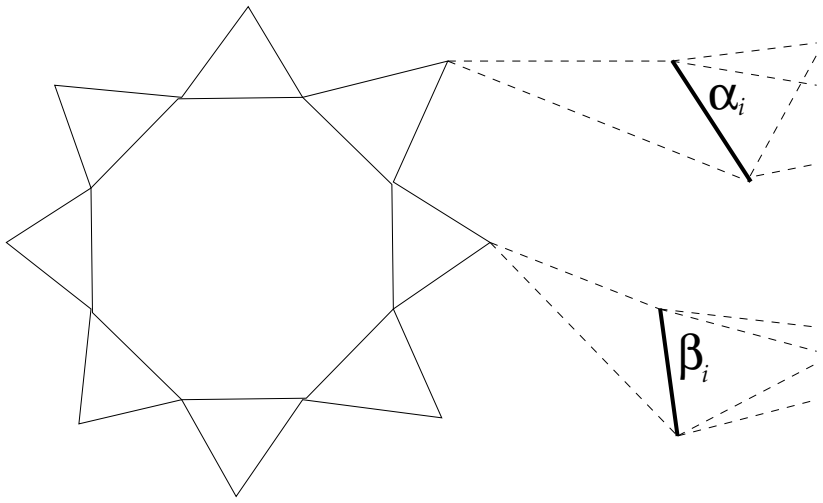

Figure 7.2: Part of the graph G constructed from the hypergraph H, showing a flower, its petals, and an α, β edge pair.

according to an arbitrary cyclic ordering of the vertices as $1, 2, \ldots, 2s_i$. One can then pick two distinct collections of s_i vertex-disjoint triangles in the graph F_i by picking either all the triangles containing the petal vertices in O_i or all those containing the petal vertices in E_i — these collections are accordingly called *odd* and *even* collections respectively. The choice of one of these collections will capture which one of the two colors given to the vertex v_i—this is the crux of the approach guiding the reduction. Now, corresponding to each hyperedge $e_j = (v_{j_1}, v_{j_2}, v_{j_3})$, we create two independent edges α_j, β_j in G. We add an edge from each endpoint of one of them, say α_j, to the vertex in O_{j_1} that corresponds to the occurrence of v_{j_1} in e_j. Recall that there are s_{j_1} vertices in O_{j_1} so a different one of them will be used for each connection corresponding to each of the s_{j_1} different hyperedges containing v_{j_1}. We make similar connections between the endpoints of α_j and

appropriate vertices of O_{j_2} and O_{j_3}. The endpoints of the second edge β_j are similarly connected to appropriate vertices in the *even* petal sets E_{j_1}, E_{j_2}, and E_{j_3}.

Denote by N the total number of vertices in G: clearly $N = \sum_{i=1}^{n} 4s_i + 4m = 16m$. By construction, G is 4-regular and therefore the number of edges in G, denoted by M, is $2N$—the crucial point is that G is sparse and $M = O(N)$. Finally, we construct an instance of MINDISAGREE on a complete graph on N vertices by labeling all edges in G as positive and the remaining edges as negative — let us denote by $\mathcal{I}$ the resulting ± 1-weighted copy of K_N. This completes our reduction, and clearly the transformation from the 3-uniform hypergraph H to $\mathcal{I}$ can be computed in polynomial time.

Consider any clustering, call it $\mathcal{C}$, of the vertices of $\mathcal{I}$, or equivalently of G. Let the *value* of a cluster be the number of edges of G within the cluster minus the number of non-edges of G within the cluster—that is, the correlation associated with edges inside the cluster. Define the value of the clustering $\mathcal{C}$, denoted $\mathsf{value}(\mathcal{C})$, to be the sum of the values of all the clusters in $\mathcal{C}$. It is easy to verify that the number of disagreements (or mistakes) in the clustering $\mathcal{C}$, denote it $\mathsf{DisAg}(\mathcal{C})$, satisfies $\mathsf{DisAg}(\mathcal{C}) = M - \mathsf{value}(\mathcal{C})$.

We now define the value $\mathsf{val}_{\mathcal{C}}(v)$ of a vertex v, with respect to the clustering $\mathcal{C}$, to be the value of the cluster containing v divided by the number of vertices in that cluster. This way the value of a cluster is equally divided among its constituent vertices. For example, if a vertex is in a singleton cluster, its value is 0, if it is in an *edge* cluster, its value is $1/2$, if it belongs to a triangle cluster, its value is 1, and so on. Note that $\mathsf{value}(\mathcal{C})$ equals the sum of the values (under $\mathcal{C}$) of all the vertices.

(i) H is 2-colorable

We first claim that if H is 2-colorable, then there is a clustering $\mathcal{C}^*$ of G in which every vertex has value 1, and therefore $\mathsf{value}(\mathcal{C}^*) = N$. In what follows, a *diamond* refers to the complete graph K_4 on four vertices minus one edge. Let $f : V \to \{\mathsf{Red}, \mathsf{Blue}\}$ be a 2-coloring under which every hyperedge of H is bichromatic. First, we pick the following clusters. For each flower structure F_i, we pick the s_i triangles of the *odd* collection (those containing the vertices in O_i) if $f(v_i) = \mathsf{Red}$, and those belonging to the *even* collection (the ones containing the vertices in E_i) if $f(v_i) = \mathsf{Blue}$. We know each hyperedge e_j is bichromatic, so assume for definiteness that two of its vertices v_{j_1}, v_{j_2} are colored Red and the third one v_{j_3} is colored Blue. Then, for this j, we pick two clusters, one a triangle containing the edge α_j together with its neighbor in O_{j_3}, and the other a diamond containing the edge β_j together with its neighbors in E_{j_1} and E_{j_2}.

It is easy to check that the clustering $\mathcal{C}^*$ defined above covers all the vertices of G. Since each vertex of G is in either a triangle or a diamond cluster, it has a value of 1 and $\mathsf{value}(\mathcal{C}^*) = N$, as claimed.

(ii) H has at least γ fraction of edges monochromatic

We now wish to argue that if every 2-coloring of H leaves γm hyperedges monochromatic, then *every* clustering $\mathcal{C}'$ of G must have value at most $(1 - \delta)N$ for some $\delta > 0$. The following claim is crucial to understanding how good clusterings (those with large value) of G must appear.

Claim: *In any clustering of $\mathcal{C}$ of G, the value of every vertex is at most 1, and if $\mathsf{val}_{\mathcal{C}}(v) = 1$,*

then v must belong to a cluster which is either a triangle or a diamond. Moreover, the supremum $(1-\rho)$ of the non-triangle and non-diamond vertex values is strictly less than 1.

The claim can be proved by straightforward inspection of the structure of the graph G since it is so sparsely connected—we omit the details. The claim asserts that $\rho > 0$; in fact one can show that $\rho = 0.2$, but all we require is that ρ is a strictly positive constant.

Now suppose there exists a clustering $\mathcal{C}'$ with $\mathsf{value}(\mathcal{C}') = (1-\delta)N$. A simple counting argument shows that we must have at least

$$n - \delta N/\rho = n - 16\delta m/\rho$$

values of i for which *every* vertex in the flower structure F_i has value equal to 1. Call the vertex $v_i \in V$ for each such i *good*. Also call an hyperedge of H good if *all* three of its vertices are good. Since there are at most $16\delta m/\rho$ bad vertices in V, there are at most $16\delta mB/\rho$ bad hyperedges.

Suppose we could prove that there is a 2-coloring of H under which every good hyperedge is bichromatic, then, since every 2-coloring of H leaves at least γm monochromatic hyperedges, we would have $16\delta B/\rho \geq \gamma$. As a consequence,

$$\mathsf{value}(\mathcal{C}') = (1-\delta)N \leq (1-\zeta)N\,,$$

where $\zeta = \rho\gamma/(16B)$, and there would be a gap of N versus $(1-\zeta)N$ for the value of the best clustering in the two cases. Recalling that

$$\mathsf{DisAg}(\mathcal{C}) = M - \mathsf{value}(\mathcal{C}) = 2N - \mathsf{value}(\mathcal{C})\,,$$

we would get a gap of N versus $(1+\zeta)N$ for the number of disagreements in the best clustering. Since $\zeta > 0$ this will prove the theorem.

Therefore it only remains to prove that there is a 2-coloring g of H under which every *good* hyperedge is bichromatic. Consider a good vertex v_i: we know all internal cycle vertices in the flower structure F_i have value 1. Since there is no diamond structure containing any of these vertices, the claim tells us they must all be covered by vertex-disjoint triangles. There are only two ways to achieve this: either the triangles containing the odd petals O_i are picked, or those containing the even petals E_i are picked. We set $g(v_i) = \mathsf{Red}$ in the former case and $g(v_i) = \mathsf{Blue}$ in the latter case (the colors given to the bad vertices are of no concern). We now prove that every good hyperedge is bichromatic under this coloring. Indeed, let e_j be a hyperedge on three good vertices $v_{j_1}, v_{j_2}, v_{j_3}$, and suppose all of them are colored Red under g. Let $w_1 \in E_{j_1}$ be the vertex that is adjacent to the endpoints of β_j. Since $\mathsf{val}_{\mathcal{C}'}(w_1) = 1$, w_1 must be clustered together with the edge β_j. The same holds for the analogous vertices w_2, w_3 from E_{j_2} and E_{j_3} respectively. But now w_1 belongs to a cluster that contains at least five elements (namely the endpoints of β_j and w_1, w_2, w_3) and therefore w_1 cannot have value 1, a contradiction. We conclude that all good hyperedges are bichromatic under g and the proof is complete. ■

Chapter 8

Efficient approximation algorithms

Throughout this dissertation we have been considering approximation algorithms that run in polynomial time. As we mentioned in Chapter 1, the number of data items in a clustering problem could be very large. For this reason, we not only seek algorithms that run in polynomial time, but those that are efficient on large data sets. To that end, we open up some of the black boxes—the linear and semidefinite program solvers—that we used in our algorithms, to see if their running times can be improved.

8.1 The MAXCORR SDP

We first consider the process of finding an optimal solution to (6.4), which can be expressed as

$$\begin{aligned} &\text{maximize} && A \bullet X \\ &\text{subject to} && x_{ii} = 1 \quad \text{for all } i \\ & && X \succeq 0\,, \end{aligned} \tag{8.1}$$

where $A \bullet X = \sum_{ij} a_{ij}x_{ij}$, the trace of AX.

It will be more convenient to deal with the following program, which we show is equivalent to (8.1). For reasons that will become clear later, we use matrix B instead of A, where $B = A + bI$ for some $b > 0$ to be specified later.

$$\begin{aligned} \text{minimize} \quad & z \\ \text{subject to} \quad & y_{ii} \leq z \quad \text{for all } i \\ & B \bullet Y = 1 \\ & Y \succeq 0\,, \end{aligned} \tag{8.2}$$

Given a feasible solution X to (8.1), for which $B \bullet X > 0$, we construct a solution Y to (8.2) by letting $Y = X/(B \bullet X)$. Similarly, we may assume that any feasible solution (Y, z) to (8.2) has $y_{ii} = z$, as B's diagonal terms are nonnegative; hence $X = Y/z$.

We can express this optimization problem as a type of game for which Freund and Schapire [31] provided an efficient algorithm. Arora, Hazan, and Kale [5] used this game framework to implement efficiently an $O(\sqrt{\log n})$-approximation algorithm for sparsest cut.

The learner (player) has a choice of n pure strategies labeled 1 through n. The environment (player) chooses a strategy $Y \in \mathcal{Y} = \{B \bullet Y = 1, Y \succeq 0\}$. The payoff that the learner gains (and thus the loss that the environment incurs) with pure strategy i, when the environment chooses Y, is y_{ii}. If we let P stand for a family of probability distributions on the set $\{1, 2, \ldots, n\}$, then the value of the game is

$$\min_{Y \in \mathcal{Y}} \max_{p \in P} \sum_{i=1}^{n} p_i y_{ii} \leq \min_{Y \in \mathcal{Y}} \max_{i} y_{ii} = z^*\,, \tag{8.3}$$

where z^* is the optimum of (8.2).

The Freund-Schapire game assumes that the environment can choose a mixed strategy Y that is maximally adversarial. This best choice is achieved by solving

$$\min_{Y \in \mathcal{Y}} \sum_{i=1}^{n} p_i y_{ii} \,, \tag{8.4}$$

where p is the learner's chosen mixed strategy. Klein and Lu [55] claim that there is an optimal solution to (8.4) that is a rank one matrix (that is, of the form vv^T for some appropriate vector v). They describe the power method [55, Section 4.3], which finds a vector whose rank one matrix is a factor $(1 + \varepsilon)$ worse than the optimal (rank one) solution. The initial claim is not, however, fully justified in Klein and Lu's text, so we provide the details here.

Lemma 8.1 *There exists a rank one optimal solution to program* (8.4).

Proof: Since the matrix Y is positive semidefinite, it may be written as the weighted sum of matrices formed by orthonormal vectors: the weights are the nonnegative eigenvalues of Y. So consider an optimal solution Y^* of the form

$$\sum_{j=1}^{n} \lambda_j w^{(j)} w^{(j)T} \,,$$

whose objective value (8.4) is $\sum_j \lambda_j s_j$, where $s_j = \sum_i p_i w_i^{(j)2}$. If we let $t_j = w^{(j)T} B w^{(j)}$, then $\sum_j \lambda_j t_j = 1$, as $B \bullet Y^* = 1$.

Imagine that for some value k, t_k were negative in the optimal solution Y^*. The matrix

$$Y' = \frac{Y - \lambda_k w^{(k)} w^{(k)T}}{1 - \lambda_k t_k}$$

is clearly in $\mathcal{Y}$. Moreover, its objective value

$$\sum_i p_i y'_{ii} = \frac{(\sum_{j=1}^{n} \lambda_j s_j) - \lambda_k s_k}{1 - \lambda_k t_k}$$

is strictly less than $\sum_{j=1}^{n} \lambda_j s_j$, as $t_k < 0$ and the other terms are all nonnegative. Consequently Y^* would not have been the optimal solution to (8.4).

The rank-one matrices formed from the vectors $w^{(j)}/\sqrt{t_j}$ are also in $\mathcal{Y}$ and have objective values s_j/t_j. Since all the t_j are positive (we can ignore j for which $s_j = t_j = 0$), it is easy to show that the largest value of s_j/t_j must be at least $\sum_j \lambda_j s_j$ and therefore some rank-one solution is optimal. ■

The Freund-Schapire game assumes that all of the y_{ii} values lie in the range $[0, 1]$. The upperbound for the (observed) payoffs to the environment is called the width (w). Had we just used the matrix A it might not have been easy to place a bound on the width, as we do below.

Lemma 8.2 *The width of the game is at most* $1/b$.

Proof: Since $Y \in \mathcal{Y}$,

$$A \bullet Y + b \sum_i y_{ii} = B \bullet Y = 1 \,.$$

Now, we claim that for the actual matrices Y that the maximally adversarial environment would choose, $A \bullet Y \geq 0$. If not, then the environment might instead have chosen

$$Y' = \frac{\text{diag}(y_{11}, \ldots, y_{nn})}{1 - A \bullet Y} \,,$$

which has $B \bullet Y' = 1$, but whose objective value (8.4) would be a fraction $1/(1 - A \bullet Y) < 1$ of Y's. Therefore we can assume that $A \bullet Y \geq 0$ and thus $\sum_i y_{ii} \leq 1/b$, implying that each (nonnegative) y_{ii} is at most $1/b$. ■

The key point now is to connect the approximation behavior of the two programs. Consider a $(1 + \varepsilon)$-approximate solution (Y, z) to (8.2), with accompanying X, and an

optimal solution (Y^*, z^*) with accompanying optimal X^*. Let L denote a lower bound for $A \bullet X^*$. The two solutions for (8.1) are related thus:

$$\begin{aligned} A \bullet X &= B \bullet X - bn \\ &\geq \frac{1}{z} - bn \\ &\geq \frac{1}{(1+\varepsilon)z^*} - bn \\ &= \frac{1}{1+\varepsilon}(\frac{1}{z^*} - bn(1+\varepsilon)) \\ &= \frac{1}{1+\varepsilon}(A \bullet X^* - bn\varepsilon)\,, \end{aligned}$$

and if we set $b = L/n$,

$$\begin{aligned} &\geq \frac{1-\varepsilon}{1+\varepsilon} A \bullet X^* \\ &\geq (1-2\varepsilon) A \bullet X^* \end{aligned}$$

This tells us that a $(1+\varepsilon)$-approximate solution to the second program (8.2) will result in a $(1-2\varepsilon)$-approximate solution to the SDP (6.4). So we now concentrate on an efficient solution to (8.2).

Now that we know we can use Klein and Lu's power method [55], we analyze the running time of the Freund-Schapire algorithm. The algorithm consists of a sequence of rounds in which the learner and the environment take turns. After T rounds we can guarantee that the average of the environment strategies $\bar{Y}$ has

$$\max_{p \in P} \sum_i p_i \bar{y}_{ii} \leq (1+\delta) z^* + \frac{w \ln n}{\delta T}\,,$$

where $1+\delta$ is the rescaling factor in the multiplicative weights algorithm of Freund and Schapire [31]. We pause to mention that we can easily obtain a vector v (to be used in

ApproxMaxQP) that satisfies $vv^T = \bar{Y}$ from the the individual vector solutions to the (8.4) programs [55]. We let $\delta = \varepsilon/4$ and

$$T = \frac{16w \ln n}{\varepsilon^2 z^*}, \tag{8.5}$$

and note that

$$\frac{1}{z^*} = A \bullet X^* + bn \leq 2A \bullet X^*.$$

Imagine for the moment that we had a lower bound L that was at least half the optimum SDP value $A \bullet X^*$. Then, substituting all these values into (8.5), we would know that after $64n \ln n/\varepsilon^2$ rounds of the game we would have a $(1-\varepsilon)$ approximate solution to the SDP. We do not have such a lower bound, but Lemma 6.6 tells us that $A \bullet X^*$ is at least $\ell = \sum_{ij} |a_{ij}|/n$. So we run the game for the prescribed number of rounds, first assuming $L = \ell$, then 2ℓ, then 4ℓ, and so forth, until we reach the obvious upper bound of $\sum_{ij} |a_{ij}|$. We then have $\log n$ potential solutions and we simply return the best of these. Note that we are sure that one of these values of L lies between $A \bullet X^*/2$ and $A \bullet X^*$ and thus are sure to obtain a $1-\varepsilon$ approximate solution.

Since we are running $\log n$ games, each for $\tilde{O}(n)$ iterations, and each execution of the power method costs $O(m \log n/\varepsilon)$ [55], the total time taken is at most $\tilde{O}(mn)$. This is clearly better than the $\tilde{O}(n^{3.5})$ guarantee of interior point methods, especially when the matrix A is sparse.

8.2 MINDISAGREE in complete instances

Recall the NEPPC formulation of MINDISAGREE in complete instances (4.6). It bears some resemblance to the dual of the maximum multicommodity flow problem, in which

each negative edge corresponds to a source-sink pair. Both the x_{ij} values for positive edges and the x'_{ij} values for negative edges now represent the lengths of these edges. We therefore wish to minimize the *total* length of all the edges subject to the constraint that the shortest NEPPC through each negative edge has length at least one. Garg and Könemann [34] have a family of efficient algorithms for multicommodity flow problems that make use of this dual length-minimization (fractional multicut) problem. We now show that our MINDISAGREE LP (4.6) can be made to fit into their framework.

They key issue with our formulation is that the source and sink vertices are identical and that the paths under scrutiny are actually cycles. We work around this by adding new nodes to the graph. We label the two endpoints of the ith negative edge s_i (source) and t_i (sink), where the numbering and orientation is arbitrary. For each pair (s_i, t_i) we create a new vertex r_i and make a new edge (r_i, t_i); there are no other edges to the new sink r_i. We then remove all of the negative edges and replace each NEPPC constraint with a path constraint from s_i to r_i of the same form. Since all paths from s_i to r_i must include (t_i, r_i) it performs the same role as the negative edge (s_i, t_i) in the constraints. Moreover, no s_j to r_j path for $j \neq i$ would use the (t_i, r_i) edge as it is a dead end.

We now have a (dual) maximum multicommodity flow problem in exactly the right form for Garg and Könemann's algorithm [34]. The running time is $O(S\, m \log m)$, where S is the amount of time needed to solve the appropriate shortest path problem, assuming that we would like a $(1+\varepsilon)$ approximate algorithm, for fixed ε. Their procedure alternates between finding the shortest source-sink path, and increasing the lengths of every each on this path. In our instance there might be $\Omega(m)$ such source-sink pairs to consider and $\Omega(n)$ edge updates between each query. Note however, that we do not need to include

the (t_i, r_i) edges directly in any shortest path procedure, as there is only one edge to each r_i.

At first glance, it seems that some sort of dynamic shortest path algorithm [26, 27] should help. However, the fact that we are making up to n updates to the edge lengths between each shortest path calculation makes these methods less efficient than the static all pairs Fibonacci heap method, which has S in $O(mn + n^2 \log n)$. Therefore our best bound on the running time for solving the NEPPC LP (4.6) is $\tilde{O}(n^5)$, since our instance is complete.

As a quick comparison, the running time of interior point LP methods is $O(N^3 L)$, where N is the number of variables and L is a parameter that reflects the size of the linear program [79]. For our purposes N is $O(n^2)$, as there is one variable for each edge, and L is $O(n^3)$, reflecting the large number of triangle inequality constraints in (4.2). So it seems that transforming the problem to the NEPPC formulation achieves a significant gain in running time over the $O(n^8)$ standard technique. We should keep in mind, however, that n^5 is still an impractical bound for large data sets.

Chapter 9

What's next

9.1 Improving the algorithms

The obvious first extension to this dissertation would be to close some of the gaps between the best-known approximation and hardness factors. Given the lack of progress on the minimum multicut problem, it seems that improving on the $O(\log n)$ region growing for MINDISAGREE in general graphs would be difficult. It might be more appropriate to look at the complete graph version, for which we currently have a factor 4 algorithm, but an integrality gap of only 2. The similarity to a feedback edge set problem that already has a 2-approximation is a promising sign. We should keep in mind, however, that solving the linear program (4.6) is inefficient, so perhaps a combinatorial algorithm, more similar to that of Bansal *et al.* [7], might be a more practical idea.

For MAXAGREE, complete instances are settled, but for general instances it might be worthwhile looking for an approach that does not involve a linear or semidefinite program. An SDP that involves the triangle inequalities does not have such a poor inte-

grality gap, but the fact remains that Swamy's SDP-based algorithm [72] is hardly better than ours. Since the MAXAGREE SDP (5.1) is similar to the MAXQP SDP (6.4), it would be good to know that an efficient implementation is available. The obvious upper and lower bounds for (5.1) differ by a factor of 2, removing a factor of $\log n$ from the running time. Nevertheless, we must check whether the $X \geq 0$ constraint, representing $v_i \cdot v_j \geq 0$, is compatible with the proof technique we used in Chapter 8.

Preliminary investigation suggests that ALGCOMPLETE does not yield anything for MAXCORR on complete instances. If we restrict ourselves to general instances, we would like to know whether a constant factor approximation for MAXQP is possible. Certainly, it will not involve the SDP (6.4), as Alon *et al.* [1] have recently shown that the integrality gap is in $O(1/\log n)$. In proving this fact, they work with a variant of the dual characterization discussed in Section 6.5.

Since the hardness results of Håstad and Venkatesh [44] do not apply, we would like to prove hardness results for the problem of maximizing the gain of max cut (the advantage over a random assignment). In light of our $\Omega(\log(1/\delta))$ approximation, obtaining an $o(1)$ hardness of approximation result would be quite challenging.

Our combinatorial algorithm for the asymmetric weighted k-center problem is essentially the best possible and it runs in $\tilde{O}(n^2)$ time, so further effort does not seem warranted.

Little is known about the pattern rotation problem. Expert opinion is that it is likely to be NP-hard, but also that proving this would be very tricky. With its special structure, we hope that an $o(\log n)$ approximation algorithm can be found, better than the greedy approach.

9.2 Consensus clustering

Robustness is a desirable feature of any clustering procedure. We would like to feel that the partitioning we obtain is not dependent on the specific procedure we use and that the clusters will not change much even if there are small errors in the data collection. Alternatively, we might wish to compensate for the variation in experimental conditions of some biological procedure, even if the clustering algorithm that is used is the same.

To this end, we would like to find a single clustering that is a mixture or average of the individual partitionings. Essentially, we want a 1-center or 1-median on the space of clusterings. The consensus clustering model assumes that we have a function to compare two clusterings. Meilă [62] has an excellent summary of the various comparison functions, including an elegant information theoretic metric. It is easy to show that for any metric comparison function the best of the original individual partitionings is a 2-approximate solution to the 1-center consensus, and a $2(1 - 1/k)$-approximate solution to the 1-median consensus, where k is the number of input partitionings. However, it seems that for the purpose of developing more sophisticated approximation algorithms, functions such as Meilă's are too complicated to be useful.

The most commonly used comparison function, the Rand distance (or Mirkin metric), counts the number of pairs of items that are in the same cluster in one of the partitionings, but not in the other. The trivial $2(1 - 1/k)$-approximation (above) is still the best known algorithm; beating this seems an obvious starting point. A more difficult variant is to do this without knowing what the original partitions are: we are only told in *how many* partitionings i and j are clustered together, for every pair ij [18].

Finally, a note of warning about the 1-center criterion. The optimum 1-center consensus might not place two items in the same cluster even if they are together in each of the input clusterings, surely an undesirable property.

9.3 Epilogue

One should keep in mind that there might be some gain in keeping on hand several different views of how to cluster various objects: the consensus might hide some informative individual opinions. Moreover, not every data set is ripe for clustering, and clustering has some inherent inconsistencies [56]. Stereotyping has its limits.

Bibliography

[1] N. Alon, K. Makarychev, Y. Makarychev, and A. Naor. Quadratic forms on graphs. Preprint.

[2] N. Alon and A. Naor. Approximating the cut-norm via Grothendieck's inequality. In *Proc. of 36th STOC*, pages 72–80, 2004.

[3] A. Archer. Inapproximability of the asymmetric facility location and k-median problems. Unpublished manuscript available at `www.orie.cornell.edu/~aarcher/Research`, 2000.

[4] A. Archer. Two $O(\log^* k)$-approximation algorithms for the asymmetric k-center problem. In *Proc. of 8th IPCO*, pages 1–14, 2001.

[5] S. Arora, E. Hazan, and S. Kale. $O(\sqrt{\log n})$ approximation to SPARSEST CUT can be found in $\tilde{O}(n^2)$ time. In *Proc. of 45th FOCS*, 2004. To appear.

[6] V. Arya, N. Garg, R. Khandekar, A. Meyerson, K. Mungala, and V. Pandit. Local search heurisitcs for k-median and facility location problems. In *Proc. of 33rd STOC*, pages 21–9, 2001.

[7] N. Bansal, A. Blum, and S. Chawla. Correlation clustering. In *Proc. of 43rd FOCS*, pages 238–47, 2002.

[8] N. Bansal, A. Blum, and S. Chawla. Correlation clustering. *Machine Learning*, 56:89–113, 2004.

[9] S. Becker, S. Thrun, and K. Obermayer, editors. *Advances in Neural Information Processing Systems 15*. MIT, 2003.

[10] A. Ben-Dor, R. Shamir, and Z. Yakhini. Clustering gene expression patterns. *J. Comp. Biol.*, 6:281–97, 1999.

[11] R. Bhatia, S. Guha, S. Khuller, and Y. Sussmann. Facility location with dynamic distance function. In *Proc. of 6th Scand. Workshop on Alg. Th. (SWAT)*, pages 23–34, 1998.

[12] M. Bilenko and R. Mooney. Learning to combine trained distance metrics for duplicate detection in databases. Technical Report AI 02-296, UT Austin, 2002.

[13] G. Calinescu, H. Karloff, and Y. Rabani. An improved approximation algorithm for multiway cut. *JCSS*, 60:564–74, 2000.

[14] M. Charikar, V. Guruswami, and A. Wirth. Clustering with qualitative information. In *Proc. of 44th FOCS*, pages 524–33, 2003.

[15] M. Charikar, S. Khuller, D. Mount, and G. Narasimhan. Algorithms for facility location problems with outliers. In *Proc. of 12th SODA*, pages 642–51, 2001.

[16] M. Charikar and A. Wirth. Maximizing quadratic programs: Extending Grothendieck's inequality. In *Proc. of 45th FOCS*, 2004. To appear.

[17] S. Chaudhuri, N. Garg, and R. Ravi. The p-neighbor k-center problem. *Info. Proc. Lett.*, 65:131–4, 1998.

[18] S. Chawla. Personal Communication, 2004.

[19] Z. Chen, T. Jiang, and G. Lin. Computing phylogenetic roots with bounded degrees and errors. *SIAM J. Comp.*, 32:864–79, 2003.

[20] J. Chuzhoy, S. Guha, E. Halperin, G. Kortsarz, S. Khanna, and S. Naor. Asymmetric k-center is $\log^* n$-hard to approximate. In *Proc. of 36th STOC*, pages 21–7, 2004.

[21] J. Chuzhoy, S. Guha, S. Khanna, and S. Naor. Asymmetric k-center is $\log^* n$-hard to approximate. Technical Report 03-038, Elec. Coll. Comp. Complexity, 2003.

[22] W. Cohen and J. Richman. Learning to match and cluster entity names. In *SIGIR Workshop on Math./Formal Methods in IR*, 2001.

[23] T. Cormen, C. Leiserson, R. Rivest, and C. Stein. *Introduction to Algorithms*. MIT and McGraw-Hill, 2001.

[24] S. Dasgupta. Performance guarantees for hierarchical clustering. In *Proc. of 15th COLT*, pages 351–63, 2002.

[25] E. Demaine and N. Immorlica. Correlation clustering with partial information. In *Proc. of 6th APPROX*, pages 1–13, 2003.

[26] C. Demetrescu and F. Italiano. Fully dynamic all pairs shortest paths with real edge weights. In *Proc. of 42nd FOCS*, pages 260–7, 2001.

[27] C. Demetrescu and F. Italiano. A new approach to dynamic all pairs shortest paths. In *Proc. of 35th STOC*, pages 159–66, 2003.

[28] D. Emanuel and A. Fiat. Correlation clustering—minimizing disagreements on arbitrary weighted graphs. In *Proc. of 11th ESA*, pages 208–20, 2003.

[29] G. Even, J. Naor, B. Schieber, and L. Zosin. Approximating minimum subset feedback sets in undirected graphs with applications. *SIAM J. Disc. Math.*, 25:255–67, 2000.

[30] U. Feige and M. Langberg. The RPR^2 rounding technique for semidefinite programs. In *Proc. of 28th ICALP*, pages 213–24, 2001.

[31] Y. Freund and R. Schapire. Adaptive game playing using multiplicative weights. *Games and Economic Behavior*, 29:79–103, 1999.

[32] A. Frieze and M. Jerrum. Improved approximation algorithms for MAX k-CUT and MAX BISECTION. In *Proc. of 4th IPCO*, pages 1–13, 1995.

[33] M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[34] N. Garg and J. Könemann. Faster and simpler algorithms for multicommodity flow and other fractional packing problems. In *Proc. of 39th FOCS*, pages 300–9, 1998.

[35] N. Garg, V. Vazirani, and M. Yannakakis. Approximate max-flow min-(multi)cut theorems and their applications. *SIAM J. Comp.*, 25:235–51, 1996.

[36] N. Garg, V. Vazirani, and M. Yannakakis. Primal-dual approximation algorithms for integral flow and multicut in trees. *Algorithmica*, 18:3–20, 1997.

[37] M. Goemans and D. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite progamming. *JACM*, 42:1115–45, 1995.

[38] I. Gørtz and A. Wirth. Asymmetry in k-center variants. In *Proc. of 6th APPROX*, pages 59–70, 2003.

[39] J. Gramm, J. Guo, F. Hüffner, and R. Niedermeier. Graph-modeled data clustering: Fixed-parameter algorithms for clique generation. In *Proc. of 5th CIAC*, pages 108–19, 2003.

[40] A. Grothendieck. Résumé de la théorie métrique des produits tensoriels topologiques. *Bol. Soc. Mat. Sao Paulo*, 8:1–79, 1953.

[41] V. Guruswami. Inapproximability results for set splitting and satisfiability problems with no mixed clauses. *Algorithmica*, 38:451–69, 2003.

[42] E. Halperin, G. Kortsarz, and R. Krauthgamer. Tight lower bounds for the asymmetric k-center problem. Technical Report 03-035, Elec. Coll. Comp. Complexity, 2003.

[43] J. Håstad. Some optimal inapproximability results. *JACM*, 48:798–859, 2001.

[44] J. Håstad and S. Venkatesh. On the advantage over a random assignment. In *Proc. of 34th STOC*, pages 43–52, 2002.

[45] D. Hochbaum, editor. *Approximation Algorithms for NP-Hard Problems*. PWS, 1997.

[46] D. Hochbaum and D. Shmoys. A best possible approximation algorithm for the k-center problem. *Math. Oper. Res.*, 10:180–4, 1985.

[47] D. Hochbaum and D. Shmoys. A unified approach to approximation algorithms for bottleneck problems. *JACM*, 33:533–50, 1986.

[48] W. Hsu and G. Nemhauser. Easy and hard bottelneck location problems. *Disc. Appl. Math.*, 1:209–16, 1979.

[49] D. Karger, P. Klein, C. Stein, M. Thorup, and N. Young. Rounding algorithms for a geometric embedding of minimum multiway cut. In *Proc. of 31st STOC*, pages 668–78, 1999.

[50] D. Karger, R. Motwani, and M. Sudan. Approximate graph coloring by semidefinite programming. *JACM*, 45:246–65, 1998.

[51] O. Kariv and S. Hakimi. An algorithmic approach to network location problems. I. The p-centers. *SIAM J. Appl. Math.*, 37:513–38, 1979.

[52] M. Kearns, R. Schapire, and L. Sellie. Toward efficient agnostic learning. *Machine Learning*, 17:115–42, 1994.

[53] S. Khot. On the power of unique 2-prover 1-round games. In *Proc. of 34th STOC*, pages 767–75, 2002.

[54] S. Khuller, R. Pless, and Y. Sussmann. Fault tolerant k-center problems. *Theor. Comp. Sci.*, 242:237–45, 2000.

[55] P. Klein and H. Lu. Efficient approximation algorithms for semidefinite programs arising from MAX CUT and COLORING. In *Proc. of 28th STOC*, pages 338–47, 1996.

[56] J. Kleinberg. An impossibility theorem for clustering. In Becker et al. [9], pages 446–53.

[57] J. Krivine. Sur la constante de Grothendieck. *C. R. Acad. Sci. Paris Ser. A-B*, 284:445–6, 1977.

[58] L. Lovasz. On the Shannon capacity of a graph. *IEEE Tr. Info. Theory*, IT-25:1–7, 1979.

[59] M. Mahdian, Y. Ye, and J. Zhang. Improved approximation algorithms for metric facility location problems. In *Proc. of 5th APPROX*, pages 229–42, 2002.

[60] A. McCallum and B. Wellner. Toward conditional models of identity uncertainty with application to proper noun coreference. In *IJCAI Workshop on Info. Integration on the Web*, 2003.

[61] A. Megretski. Relaxation of quadratic programs in operator theory and system analysis. In *Systems, Approximation, Singular Integral Operators, and Related Topics (Bordeaux, 2000)*, pages 365–92. Birkhäuser, Basel, 2001.

[62] M. Meilă. Comparing clusterings by the variation of information. In *Proc. of 16th COLT*, pages 173–87, 2003.

[63] S. Monti, P. Tamayo, J. Mesirov, and T. Golub. Consensus clustering: A resampling-based method for class discovery and visualization of gene expression microarray data. *Machine Learning*, 52:91–118, 2003.

[64] A. Nemirovski, C. Roos, and T. Terlaky. On maximization of quadratic form over intersection of ellipsoids with common center. *Math. Prog. Ser. A*, 86:463–73, 1999.

[65] Y. Nesterov. Global quadratic optimization via conic relaxation. Technical Report DP-9860, Center Op. Res. Econometrics, Univ. Cath. Louvain, 1998.

[66] R. Panigrahy and S. Vishwanathan. An $O(\log^* n)$ approximation algorithm for the asymmetric p-center problem. *J. Algorithms*, 27:259–68, 1998.

[67] C. Papadimitriou. *Computational Complexity*. Addison Wesley Longman, 1994.

[68] J. Plesnik. A heuristic for the p-center problem in graphs. *Disc. Appl. Math.*, 17:263–268, 1987.

[69] E. Rietz. A proof of the Grothendieck inequality. *Israel J. Math.*, 19:271–6, 1974.

[70] F. Roberts. Discrete mathematics. In *International Encyclopedia of the Social and Behavioral Sciences*, pages 3743–6. Elsevier, 2001.

[71] R. Shamir, R. Sharan, and D. Tsur. Cluster graph modification problems. In *Proc. of 28th Workshop on Graph Theory (WG)*, pages 379–90, 2002.

[72] C. Swamy. Correlation Clustering: Maximizing agreements via semidefinite programming. In *Proc. of 15th SODA*, pages 519–20, 2004.

[73] L. Trevisan, G. Sorkin, M. Sudan, and D. Williamson. Gadgets, approximation, and linear programming. *SIAM J. Comp.*, 29:2074–97, 2000.

[74] V. Vazirani. *Approximation Algorithms*. Springer, 2001.

[75] S. Vishwanathan. An $O(\log^* n)$ approximation algorithm for the asymmetric p-center problem. In *Proc. of 7th SODA*, pages 1–5, 1996.

[76] K. Wagstaff, C. Cardie, S. Rogers, and S. Schroedl. Constrained k-means clustering with background knowledge. In *Proc. of 18th Int. Conf. Machine Learning*, pages 577–84, 2001.

[77] D. Williamson. Lecture notes on approximation algorithms. Technical Report RC 21273, IBM Research, 1999.

[78] E. Xing, A. Ng, M. Jordan, and S. Russell. Distance metric learning with application to clustering with side-information. In Becker et al. [9], pages 505–12.

[79] Y. Ye. An $O(n^3L)$ potential reduction algorithm for linear programming. *Mathematical Programming*, 50:239–58, 1991.

[80] U. Zwick. Outward rotations: A tool for rounding solutions of semidefinite programming relaxations, with applications to max cut and other problems. In *Proc. of 31st STOC*, pages 679–87, 1999.