Abstract of "Approximation Schemes for Inferring Rankings and Clusterings from Pairwise Data" by Warren Schudy, Ph.D., Brown University, May 2011.

In correlation clustering, given similarity or dissimilarity information for all pairs of data items, the goal is to find a clustering of the items into similarity classes, with the fewest inconsistencies with the input. This problem is hard to approximate in general but we give arbitrarily good approximation algorithms (PTASs) for two interesting special cases: when there are few clusters, and when the input is generated from a natural noisy model. In the feedback arc set problem in tournaments, given comparison information (a better than b) for all pairs of data items, the goal is to find a ranking of the items with the fewest inconsistencies with the input. We give the first PTAS for this problem. We then extend our techniques to a more general class of problems called fragile dense problems.

Approximation Schemes for Inferring Rankings and Clusterings from Pairwise Data

by
Warren Schudy
B. S., Worcester Polytechnic Institute, 2005
Sc. M., Brown University, 2007

A dissertation submitted in partial fulfillment of the
requirements for the Degree of Doctor of Philosophy
in the Department of Computer Science at Brown University

Providence, Rhode Island
May 2011

This work is based on earlier works:

- How to rank with few errors, in STOC 2007, © ACM, 2007. http://doi.acm.org/10.1145/1250790.1250806
- Yet another algorithm for dense max cut: go greedy, in SODA 2008, © SIAM, 2008.
- Linear time approximation schemes for the Gale-Berlekamp game and related minimization problems, in STOC 2009, © ACM, 2009. http://doi.acm.org/10.1145/1536414.1536458
- Bounding and Comparing Methods for Correlation Clustering Beyond ILP, in ILP for NLP 2009, © ACL, 2009.
- Correlation Clustering with Noisy Input, in SODA 2010, © SIAM, 2010.

This dissertation by Warren Schudy is accepted in its present form by
the Department of Computer Science as satisfying the dissertation requirement
for the degree of Doctor of Philosophy.

Date _____ _____
Claire Mathieu, Director

Recommended to the Graduate Council

Date _____ _____
Amy Greenwald, Reader

Date _____ _____
Marek Karpinski, Reader
(University of Bonn)

Date _____ _____
Phillip Klein, Reader

Approved by the Graduate Council

Date _____ _____
Peter Weber
Dean of the Graduate School

Acknowledgements

I would also like to thank my collaborators without whom this thesis would not have happened: Micha Elsner, Amy Greenwald, Marek Karpinski, Zheng Li, and Claire Mathieu.

I would also like to thank various people who officially and unofficially guided me during my undergraduate and graduate studies, including Padmanabhan K. Aravind, Dan Dougherty, Kathi Fisler, Shriram Krishnamurthi, Claire Mathieu, George Phillies, L. Ramdas Ram-Mohan, John Savage, Meinolf Sellman, and D. Sivakumar.

Last but the opposite of least I would like to thank my friends and family who supported me and kept me sane.

Contents

1 Introduction **1**

1.1 Problems and results 1
1.2 Techniques 6
1.3 Organization 7
1.4 Other Results 7
1.5 Papers 8

2 Additive error **10**

2.1 Introduction 10
2.2 Analyzing Algorithm **??** for max cut 12
2.3 Analyzing Algorithm **??** 16
2.4 Analyzing Algorithm 2.2 24
2.5 Proof of Theorem **??** 26
2.6 Proof of Theorem **??** 26
2.7 Open problems 26

3 Fragile MIN-k-CSPs **28**

3.1 Introduction 28
3.2 Intuition 28
3.3 Model 30
3.4 Algorithm 31
3.5 Analysis of Algorithm 3.2 32
3.6 Computation of $x^{(3)}$ 36

4 Correlation Clustering with a fixed number of clusters **38**

4.1 Introduction 38
4.2 Intuition 39
4.3 Correlation and Hierarchical clustering 40
4.4 Reduction to Rigid MIN-2CSP 41
4.5 Analysis of Algorithm 4.2 43

5 Correlation Clustering with Noisy Input **47**
5.1 Introduction . . . 47
5.2 Algorithms . . . 49
5.3 Proof of Theorem 5.1 . . . 53
5.4 Proof of Theorem 5.2 . . . 64
5.5 Proof of Theorem 5.3 . . . 69

6 Correlation clustering experiments **75**
6.1 Introduction . . . 75
6.2 Algorithms . . . 75
6.3 Bounding with SDP . . . 77
6.4 Experiments . . . 78
6.5 Conclusions . . . 81

7 Feedback arc set **83**
7.1 Introduction . . . 83
7.2 Main Results . . . 86
7.3 Bounding the cost of partition-respecting rankings . . . 89
7.4 Summing errors over the recursion tree (proof of Lemma 7.7) . . . 93
7.5 Runtime . . . 95
7.6 Derandomization . . . 97
7.7 Appendix . . . 98

8 Ranking MIN-CSPs: approximation algorithms **100**
8.1 Introduction . . . 100
8.2 Intuition and main ideas . . . 102
8.3 Approximation Algorithm . . . 103
8.4 Runtime analysis . . . 106
8.5 Analysis of σ^1 . . . 106
8.6 Analysis of σ^2 . . . 108
8.7 Analysis of π^3 . . . 110
8.8 Analysis of π^4 . . . 113

9 Ranking MIN-CSPs: exact algorithms **114**
9.1 Introduction . . . 114
9.2 Algorithms and analysis . . . 115

10 Conclusions **119**

Bibliography **120**

⋆ Portions of this thesis have previously appeared elsewhere [Kenyon-Mathieu and Schudy, 2007, Mathieu and Schudy, 2008b, 2010, Karpinski and Schudy, 2009a, Elsner and Schudy, 2009]. I would like to thank my coauthors Micha Elsner, Marek Karpinski and Claire Mathieu for permission to reproduce portions of these joint works in this thesis.

Chapter 1

Introduction

We are surrounded by NP-complete problems. Delivering packages efficiently requires solution to variants of the traveling salesman problem (TSP). Problems in the design and verification of microprocessors are often reduced to the NP-complete problem of satisfiability of conjunctive normal form propositional formulae, also known as SAT. For hundreds of additional NP-complete problems see Garey and Johnson [1979].

It is well known that if there is an efficient (i.e. polynomial time) algorithm for *any* NP-complete problem then there is an efficient algorithm for *all*. Unfortunately despite decades of effort no polynomial time algorithm has been found. One way around this difficulty is *approximation algorithms*, which do not generally find the optimum solution but provably find a solution that is not much worse. An algorithm for a minimization problem is an α-approximation if its output has cost at most α times the cost of the optimum solution. A polynomial-time approx scheme (PTAS) is a family of $1+\epsilon$-approximation algorithms for all $\epsilon > 0$ with runtime polynomial in the input size m. The runtime of a PTAS need not be polynomial in $1/\epsilon$; for example $m^{O(1/\epsilon^2)}$ and $O(m) + 2^{O(1/\epsilon^2)}$ count as polynomial.

In this thesis my collaborators and I study approximation algorithms for two problems, namely *feedback arc set tournament* and *correlation clustering*, plus generalizations.

1.1 Problems and results

Feedback arc set tournament

Suppose you ran a chess tournament, everybody played everybody (a.k.a. round robin) and you wanted to use the results to rank everybody. Unless you were really lucky, the results would not be acyclic, so you could not just sort the players by who beat whom. A natural objective is to find a ranking that minimizes the number of upsets, where an upset is a pair of players where the player ranked lower in the ranking beat the player ranked higher. Minimizing the number of upsets is called *feedback arc set problem* on tournaments (FAST). The complementary problem of *maximizing* the

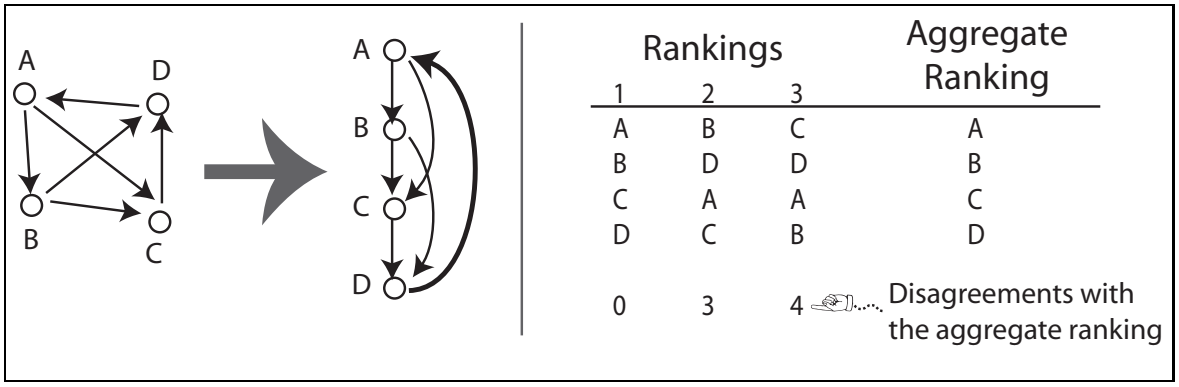

Figure 1.1: Feedback arc set tournament (left) and Kemeny rank aggregation (right) problems.

number of pairs that are *not upsets* is called the *maximum acyclic subgraph problem* on tournaments. These problems are NP-hard [Ailon et al., 2008, Alon, 2006, Charbit et al., 2007] (see also [Conitzer, 2006]). Claire Mathieu and I give the first PTAS (Theorem 7.2) for FAST.

In statistics and psychology, one motivation is *ranking by paired comparisons* [Slater, 1961]: here, you wish to sort some set by some objective but you do not have access to the objective, only a way to compare a pair and see which is greater; for example, determining people's preferences for types of food. This problem attracted computational attention as early as 1961 [Slater, 1961] (for comparison Hoare published quicksort the same year). Feedback arc set in tournament graphs and closely related problems have also been used in machine learning [Cohen et al., 1999, Ailon and Mohri, 2008].

A second application of (weighted) feedback arc set tournament is *rank aggregation*. Frequently, one has access to several rankings of objects of some sort, such as search engine outputs [Dwork et al., 2001], and desires to aggregate the input rankings into a single output ranking that is similar to all of the input rankings: it should have minimum average distance from the input rankings, for some notion of distance. This ancient problem was already studied in the context of voting by [Borda, 1781] and [Condorcet, 1785] in the 18^{th} century, and has aroused renewed interest recently [Dwork et al., 2001, Conitzer et al., 2006]. A natural notion of distance is the number of pairs of vertices that are in different orders: this defines the *Kemeny rank aggregation* problem [Kemeny, 1959, Kemeny and Snell, 1962] illustrated in Figure 1.1. This choice yields a maximum likelihood estimator for a certain naïve Bayes model [Young, 1995]. This problem is NP-hard [Bartholdi et al., 1989], even with only 4 voters [Dwork et al., 2001]. There is a randomized 4/3 approximation algorithm [Ailon et al., 2008]. Claire Mathieu and I improve on these results by giving a polynomial time approximation scheme (PTAS) (Corollary 7.3) via reduction to (a weighted version of) FAST.

The constants in our PTASs for FAST and Kemeny rank aggregation make them impractical but the role of local search in our analysis helps shed light on the excellent performance of local search for Kemeny rank aggregation observed in practice [Schalekamp and van Zuylen, 2009].

As an additional illustration of the power of our techniques we show (Theorem 9.1) that the *optimal* feedback arc set of a tournament graph can be computed in time $2^{O(\sqrt{OPT})}$, a moderate improvement on the previously best known runtime of $OPT^{O(\sqrt{OPT})}$.

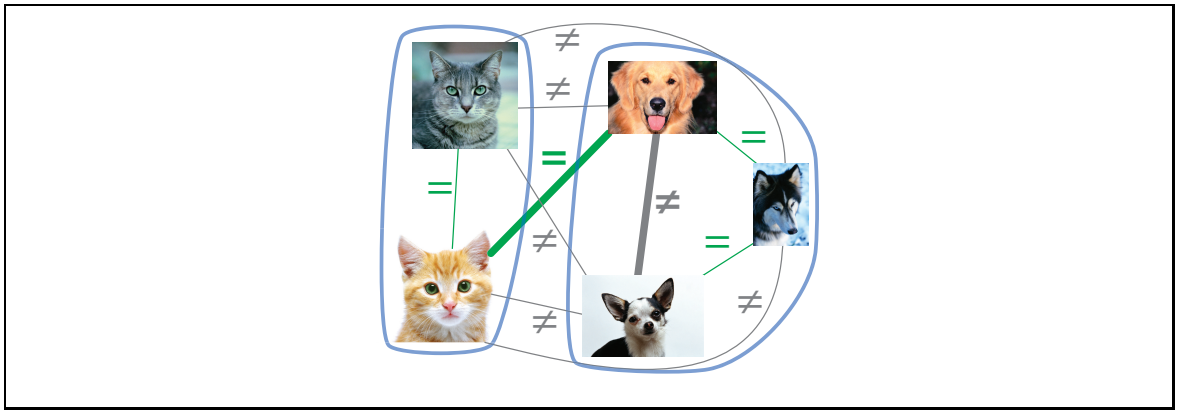

Figure 1.2: An example correlation clustering problem and optimal clustering. The optimal clustering (dogs and cats) has two disagreements, shown in bold.

Correlation Clustering

Clustering is an important tool for analyzing large data sets. Correlation clustering [Ben-Dor et al., 1999, Bansal et al., 2004] is a type of clustering that uses a very basic form of input data: indications that certain pairs of vertices (pairs of data items) belong in the same cluster, and certain other pairs belong in different clusters [Bansal et al., 2004]. Unfortunately the information is not necessarily consistent, possibly claiming for example that "cat" is similar to "dog" and "dog" is similar to "bog" but "cat" is not similar to "bog". We assume that we have information about every pair of objects. The goal is to find a clustering, that is, a partition of the vertices, that agrees with as many pieces of information as possible (maximization) or disagrees with as few as possible (minimization). Correlation clustering is illustrated in Figure 1.2. This has applications in data mining and natural language processing [Cohen and Richman, 2002, Finkel and Manning, 2008]. Correlation clustering has no PTAS unless P=NP [Charikar et al., 2005], so to get good approximation algorithms we must restrict the problem somehow to separate real-world instances from those produced by the reductions of Charikar et al. [2005].

One possible restriction is limiting the number of clusters to some small number d. With this assumption Giotis and Guruswami found a PTAS [Giotis and Guruswami, 2006], but their runtime of $n^{O(9^d/\epsilon^2)}$ leaves something to be desired. Marek Karpinski and I found a PTAS (Theorem 4.1) with runtime $n^2 2^{O((\log d)d^6/\epsilon^2)}$, which improves on Giotis and Guruswami [2006] in two ways. Firstly it is simply exponential in d rather than doubly exponential and secondly the power of n is a constant 2, independent of d and ϵ. Our result extends to the (previously known) hierarchical generalization of correlation clustering, yielding the first PTAS known for that problem.

An alternative way to work around the hardness results of Charikar et al. [2005] is to assume that the input is generated by chance rather than an adversary. Claire Mathieu and I study the following semirandom noisy model which is compromise between random and adversarial: start from an arbitrary partition of the vertices into clusters. Then, for each pair of vertices, the similarity information is corrupted (noisy) independently with probability p. Finally, an adversary generates

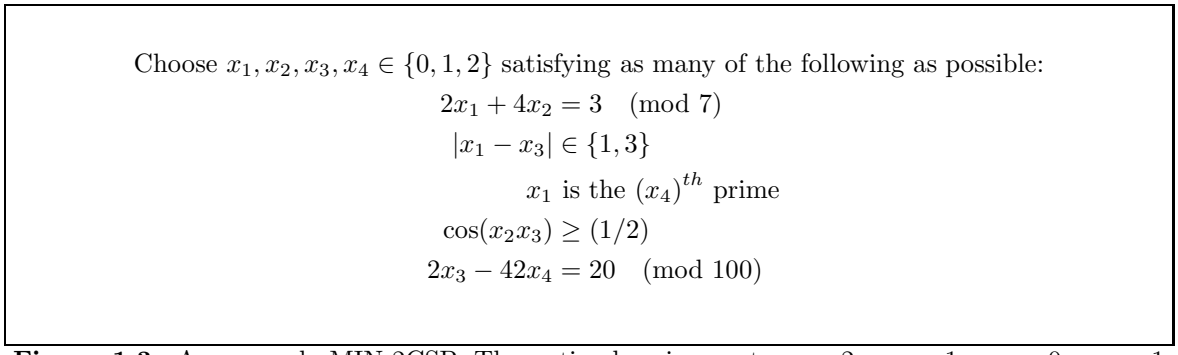

Figure 1.3: An example MIN-2CSP. The optimal assignment $x_1 = 2$, $x_2 = 1$, $x_3 = 0$, $x_4 = 1$ satisfies all constraints except the last and hence has cost 1.

the input by choosing similarity/dissimilarity information arbitrarily for each corrupted pair of vertices. In this model we give a $(1 + O(n^{-1/6}))$-approximation (Theorem 5.1), which is even better than a PTAS. The algorithm achieving this result involves solving and rounding a semi-definite programming relaxation. This semi-definite program was previously used in Charikar et al. [2005] with a different rounding method. We can also perfectly recover the planted clusters that are relatively large (Theorems 5.2 and 5.3).

Our noisy input algorithm is relatively implementable, with the time-consuming part being the solution of a semi-definite program. Micha Elsner and I tested a variety of algorithms on correlation clustering instances from two natural language processing applications (Chapter 6). The semi-definite programming relaxation produced much better lower bounds than the simple bounds used in previous work. On the other hand the clusterings found by our rounding technique were no better than clusterings found by much faster greedy and local search techniques. We conclude that the semi-definite program is useful for lower-bounds but does not appear to be competitive for finding clusterings.

Generalizations

The reader may be wondering what our results have in common and whether there are any other problems amenable to our techniques. We answer these questions by describing generalizations, first of the correlation clustering with a fixed number of clusters result and second the feedback arc set tournament result.

We now introduce the abstract problem MIN-kCSP which generalizes d-correlation clustering. An arity k constraint satisfaction problem (k-CSP) consists of a constant arity k (usually $k = 2$), a set of variables V which take values from a constant-sized domain D, and a set of constraints on the variables. Each constraint depends on a set of k of the variables and is *satisfied* by some of the possible configurations of those variables. We assume that every set of k variables is constrained by a number of constraints bounded above by a constant, which is usually 1. Satisfying all the constraints simultaneously is generally impossible so the goal of a MIN-kCSP is to minimize the number of unsatisfied constraints. Equivilantly one can maximize the number of satisfied constraints, which

is known as a MAX-kCSP. (The distinction between MIN-CSPs and MAX-CSPs is important for approximation algorithms since the optimum values can be quite different.)

We now give some examples of the expressive power of these abstract classes of problems. The d-correlation clustering problem can be reformulated as a CSP. It has $k = 2$, one variable per vertex, and $D = \{1, 2, \ldots, d\}$. An edge labeled "=" between vertices u and v corresponds to a constraint over the corresponding variables that is satisfied if the two variables are assigned the same value. Edges labeled "$\neq$" correspond to analogous constraints.

The unique games problem is another MIN-2CSP. Each constraint depends on $k = 2$ variables. The constraint over variables (u, v) is labeled with a bijection π_{uv} over the domain D. This constraint is satisfied if and only if π_{uv} maps the value of u's variable to the value of v's variable. There is a famous *unique games conjecture* on the hardness of approximating unique games, which if true would show many known approximation algorithms are the best possible.

The famous MAX CUT problem consists of finding a cut of an undirected graph with the maximum number of edges crossing the cut. This problem can be modeled as a 2CSP in a way analogous to d-correlation clustering. The minimization variant is known as MIN UNCUT. Unfortunately neither MAX CUT nor MIN UNCUT have PTASs unless P = NP Arora et al., so we cannot generalize our results to all kCSPs. So what makes correlation clustering with a fixed number of clusters different? We identify two properties: *density* and *fragility*.

We discuss three types of density. We say that an instance is *average-dense* if there are $\Omega(n^k)$ constraints; i.e. within a constant factor of the maximum possible number of constraints. We say that an instance is *everywhere-dense* if each variable participates in $\Omega(n^{k-1})$ constraints, which is again within a constant factor of the maximum possible number. We say that an instance is *fully dense* if there is a constraint for every set of k variables. Correlation clustering is fully dense. A random assignment satisfies a constant fraction of the constraints (excluding degenerate constraints that are never satisfied) so average-dense MAX-kCSPs have optimum value $\Theta(n^k)$. So-called *additive error algorithms* find an assignment with cost at most ϵn^k *plus* the optimum cost, which implies a PTAS for all MAX-kCSPs [Frieze and Kannan, 1999, Alon et al., 2002]. Claire Mathieu and I provide an additive error algorithm (Theorem 2.2) that is much simpler than previous ones (Theorem 2.2).

Additive error algorithms are insufficient for MIN-kCSPs since optimum values of minimization problems can be very small, even 0. Indeed there are many MIN-kCSPs without PTASs (unless P=NP) even on everywhere-dense instances, for example MIN-3UNCUT. There are also fully dense (albiet unatural) MIN-kCSPs with no PTAS. Our PTAS therefore cannot rely on density alone but must also rely on a specific property of correlation clustering constraints.

The key property of d-correlation clustering is that *most of the vertices in a low cost clustering cannot be moved to different clusters without increasing the cost by* $\Omega(n)$. We now give a wide class of problems that also satisfy this property. A constraint is *fragile* if modifying any variable in a satisfied constraint makes the constraint unsatisfied. A CSP is *fragile* if all of its constraints are. Several interesting CSPs feature fragile constraints, including MIN-2UNCUT, unique games, and nearest codeword problem. Marek Karpinski and I introduce the class of fragile MIN-CSPs and

show that they all satisfy this key property and hence have a PTAS (Theorem 3.1). Our runtime is essentially the best possible up to constants. The d-correlation clustering problem is not fragile but does satisfy this key property, which as noted before allows us to give a PTAS (Theorem 4.1).

Our PTAS for feedback arc set tournament (FAST) can also be generalized. A ranking CSP [Ailon and Alon, 2007] is similar to an ordinary CSP except that the goal is to choose an ordering of the objects rather than an assignment to variables. In general a ranking CSP consists of a constant arity k (2 for FAST), a set of vertices V and a set of constraints on the vertices. Each constraint depends on the ordering of a set k of vertices and is satisfied by some of the possible orderings of those vertices. In FAST each directed edge (game) corresponds to a constraint which is satisfied whenever the winner of the game is ordered before the loser. A second MIN-RANK-CSP is *betweeness*: rank objects given information of the form v is between u and w, i.e. either $u < v < w$ or $w < v < u$. Marek Karpinski and I give (Theorem **??**) a PTAS for fully dense betweenness. We also generalize this result to any weakly fragile rank CSPs (defined later).

1.2 Techniques

Our algorithms and analyses are based on a variety of techniques including:

1. **Exhaustive sampling (Chapters 2, 3 and 8)**: We make extensive use of this technique, introduced in Arora et al. [1995], for effectively taking a random sample of optimal assignments even though the optimal assignment is unknown.

2. **Local search (Chapters 7 and 8)**: Local search techniques are extensively used to solve NP-complete optimization problems in practice. Local search is difficult to analyze and therefore is rarely used in approximation algorithms. For example a standard approximation algorithms textbook [Vazirani, 2001] mentions local search only in chapter notes and exercises. We are aware of only one other state of the art approximation algorithm that uses local search: the one for the k-median problem by Arya et al. [2004]. Local search plays a key role in several of our PTASs.

3. **Martingales and Azuma-Hoeffding inequality (Chapter 2)**: We use martingales to, roughly speaking, take a random sample of a moving target.

4. **Semi-definite programming duality (Chapter 5)**: Semi-definite programming is an extension of linear programming that has attracted increasing interest in the approximation algorithms community since the seminal work of Goemans and Williamson [1995]. We analyze a semi-definite program by presenting matching primal and dual feasible solutions and using complementary slackness conditions.

5. **Spectra of random matrices (Chapter 5)**: We extend the techniques in Füredi and Komlós [1981] to analyze the eigenvalue spectra of a certain class of random symmetric matrices.

1.3 Organization

The remainder of this introductory Chapter 1 describes the organization of this thesis, describes some results done during my graduate studies that are not part of this thesis, and lists the papers where the results of this thesis were published.

The various chapters that make up this thesis can be read more or less independently of each other. The limited dependencies that exist are described below.

In chapter 2 we present known results on additive error for CSPs and our contributions [Mathieu and Schudy, 2008b] to this area. These results are used as a subroutine in many of our results.

With this foundation in hand we present our results on fragile MIN-CSPs [Karpinski and Schudy, 2009a] in section 3. This chapter uses Theorem 2.2, which is presented in Section 2.1, but only in a black-box fashion.

Chapter 4 introduces the correlation clustering and a hierarchical generalization. It then describes our PTAS for correlation clustering with a fixed number of clusters. This chapter also uses Theorem 2.2. The intuition (but not the theorems) builds on Chapter 3.

Chapter 5 studies the correlation problem again under a noisy input model. This chapter is essentially self-contained, depending only on the introduction to the correlation clustering problem from Section 4.1.

Chapter 6 describes our experimental results for correlation clustering. This chapter depends on the introduction to correlation clustering from Section 4.1. It also uses the semi-definite program described in Chapter 5.

Chapter 7 describes the feedback arc set tournament problem, Kemeny Rank Aggregation, and our PTAS for these problems. It uses Theorem 2.2 but is otherwise self-contained.

Chapter 8 describes our fragile ranking framework and the PTAS for these problems. This chapter uses the PTAS from Chapter 7 as a black box.

In Chapter 9 we give some algorithms for solving the ranking problems described in Chapters 7 and 8 exactly in exponential time.

Finally Chapter 10 concludes with some interesting open problems.

1.4 Other Results

The following results of mine are outside the scope of this thesis but are mentioned here for reference. See the papers [Greenwald et al., 2008, Schudy, 2008, Mathieu et al., 2010, Schudy, 2010] for more information.

An online decision problem (ODP) consists of a series of rounds, during each of which an agent chooses one of n pure actions and receives a reward corresponding to its choice. For example the agent may be playing a matrix game such as rock-paper-scissors or chicken repeatedly. The agent's objective is to maximize its cumulative rewards. It can work towards this goal by abiding by an online learning algorithm, which prescribes a possibly mixed action (i.e., a probability distribution over the set of pure actions) to play each round, based on past actions and their corresponding

rewards. No-internal-regret (NIR) learners converge to the set of correlated equilibria in repeated matrix games. However, standard NIR learning algorithms involve a fixed point calculation during each round of learning, which is time-consuming when the number of pure actions available to the player is large. Amy Greenwald, Zheng Li and I [Greenwald et al., 2008] give a natural NIR learner that only requires a matrix-vector multiplication, improving the runtime.

I show [Schudy, 2008] how to use $O(\log^2 n)$ reachability queries suffice to compute strongly connected components and topological sort of directed graphs. This yields faster parallel algorithms for these problems, reducing the number of processors needed to achieve a speedup of s from $O(s^3)$ to $O(s^2)$. Parallel computation of strongly connected components has applications in scientific computing [McLendon et al., 2005, 2001, Plimpton et al., 2000]. Our algorithm is simple with reasonable hidden constants and hence may be practical.

Ocan Sankur, Claire Mathieu and I have some preliminary results on online correlation clustering. In this problem the objects to be clustered arrive one by one. The algorithm may add new objects to existing clusters and possibly merge clusters, but is not allowed to split existing clusters. The goal is to maintain a clustering that is competitive with the offline optimum clustering.

A *Las Vegas* algorithm is a randomized algorithms that returns only correct answers, but has random runtime. The runtime distributions of Las Vegas algorithms are often heavy tailed, so it is often helpful to restart them periodically with a new random seed. I [Schudy, 2010] introduce a new measure of the hardness of an instance, namely the expected number of restarts r^* needed by the optimal strategy. I give a natural class of restart strategies, parameterized by the desired competitive ratio as a function of r^*, that generalizes both the restart strategy of Luby et al. [1993] and geometrically increasing timeouts. I show that this class of restart strategies is optimal up to constant factors. One member of our family of restart strategies achieves a $O((\log r^*)^{1.1})$ competitive ratio.

1.5 Papers

Papers are available from `http://www.cs.brown.edu/~ws/` or by clicking on author names below.

Feedback arc set and generalizations:

- C. Kenyon-Mathieu and W. Schudy. How to rank with few errors: a PTAS for weighted feedback arc set on tournaments. In *Procs. 39th ACM STOC*, pages 95–103, 2007.
- C. Mathieu and W. Schudy. How to rank with fewer errors – a PTAS for feedback arc set in tournaments. In Submission, 2009.
- M. Karpinski and W. Schudy. Approximation of the Betweeness-Tournament Problem and Some Ranking Generalizations. In preparation.

Correlation clustering and generalizations:

- M. Karpinski and W. Schudy. Linear time approximation schemes for the Gale-Berlekamp game and related minimization problems. In *STOC '09: Proceedings of the 41st annual ACM symposium on Theory of computing*, pages 313–322, 2009.
- M. Elsner and W. Schudy Bounding and Comparing Methods for Correlation Clustering Beyond ILP. In *Proceedings of the Workshop on Integer Linear Programming for Natural Language Processing*, pages 19–27. Association for Computational Linguistics, June 2009.
- C. Mathieu and W. Schudy. Correlation clustering with noisy input. In *Procs. 21^{st} SODA (to appear)*, 2010.

Additive error:

- C. Mathieu and W. Schudy. Yet Another Algorithm for Dense Max Cut: Go Greedy. In *Proc. 19th ACM-SIAM SODA*, pages 176–182, 2008.

Results not in thesis:

- A. Greenwald, Z. Li, and W. Schudy. More efficient internal-regret-minimizing algorithms. In *21st Annual Conference on Learning Theory - COLT 2008*, pages 239–250, 2008.
- W. Schudy. Finding strongly connected components in parallel using $O(log^2 n)$ reachability queries. In *SPAA '08: Proceedings of the twentieth annual symposium on Parallelism in algorithms and architectures*, pages 146–151, 2008.
- C. Mathieu, O. Sankur, and W. Schudy. Online correlation clustering. In *Procs. 27^{th}* STACS *(to appear)*, 2010.
- W. Schudy. Optimal restart strategies for tree search. In *NESCAI '10: New England Student Colloquim on Artificial Intelligence*, 2010.

Chapter 2

Additive error

2.1 Introduction

The results presented in this chapter are joint work with Claire Mathieu. Most of these results previously appreared in Mathieu and Schudy [2008b].

Over a decade ago two groups [Arora et al., 1995, Fernandez de la Vega, 1996] independently discovered polynomial-time approximation algorithms for MAX-CUT that produce a cut with value at least the optimum value minus ϵn^2. Such an algorithm is said to have *additive error* of ϵn^2. This yields a PTAS for average-dense MAX-CUT instances. The fastest additive error algorithms [Alon et al., 2002] have constant runtime $2^{O(1/\epsilon^2)}$ for approximating the value of any MAX-kCSP (described in Chapter 1) over a binary domain D. This can be generalized to an arbitrary domain D. To see this, note that we can code D in binary and correspondingly enlarge the arity of the constraints to $k\lceil \log |D| \rceil$. A random sample of $\tilde{O}(1/\epsilon^4)$ variables suffices to achieve an additive approximation [Alon et al., 2002, Rudelson and Vershynin, 2007]. These results extend to MAX-BISECTION [Fernandez de la Vega et al., 2006]. Ailon and Alon [2007] show that if the general version of a CSP is NP-hard to solve exactly then it is NP-hard to approximation to within an additive $n^{r-\epsilon}$ for all $\epsilon > 0$.

An additive error algorithm is a crucial subroutine in many of our PTASs. In this chapter we describe our own novel additive error algorithms, which are substantially simpler than previous algorithms. Our new techniques also allow for a slight technical improvement on previous work. The *sample complexity* of a MAX-k-CSP is the . Previously the sample complexity of MAX-k-CSPs was known to be $O(1/\epsilon^4)$ for $k = 2$ [Rudelson and Vershynin, 2007] and $\tilde{O}(1/\epsilon^4)$ for $k \geq 3$ [Alon et al., 2002]; we show that the former bound holds for $k \geq 3$ as well.

We describe our algorithms in the context of MaxCut. The greedy algorithm for MaxCut considers vertices one by one in arbitrary order and places each of them on the left or right side of the cut, depending on the number of neighbors that are already placed on each side (breaking ties arbitrarily). It is well known that the greedy algorithm is a 2-approximation, and that this bound

Take a sample S of $t_0 = 1/\epsilon^2$ variables chosen uniformly at random without replacement.
for each of the d^{t_0} possible assignments of variables S **do**
  for each variable v of $V \setminus S$ in random order, **do**
    Assign v to the value that maximizes the number of resulting satisfied constraints.
  end for
end for
Output the best assignment found.

Figure 2.1: Simplest-to-analyze greedy algorithm

Take a sample S of $t_1 = O(1/\epsilon^4)$ variables chosen uniformly at random.
Find a near-optimal assignment to the problem induced by the input on S, using Algorithm 2.1.
for each variable v of $V \setminus S$ in random order **do**
  Greedily assign a value to v, maximizing the number of resulting crossing edges.
end for

Figure 2.2: Fastest greedy algorithm

is tight if an adversary controls the input graph and the order in which the vertices are considered.

Here, we take advantage of the power of randomness by considering vertices *in random order*. We define three variants of the greedy algorithm for MaxCut. The first algorithm is closest to the "exhaustive search" techniques of previous papers, hence easiest to analyze.

Theorem 2.1. *For any $\epsilon > 0$, Algorithm 2.1 has running time $O(n^2)2^{O(1/\epsilon^2)}$ and the expected value of the output is at least $OPT - O(\epsilon n^2)$.*

Thus, for average dense instances, Algorithm 2.1 is a polynomial-time approximation scheme.

The second algorithm is basically the greedy algorithm with random order, except that we start with a good "hint" of a near-optimal solution, in the form of a near optimal solution in a constant size sample. Note that the running time separates into one term depending on ϵ but not on n (for solving the problem in the sample) and the other term depending on n but not on ϵ (for running the greedy algorithm on the rest of the graph.) Thus it is the fastest of our variants.

Theorem 2.2. *For any $\epsilon > 0$, Algorithm 2.2 has running time $O(n^2) + 2^{O(1/\epsilon^2)}$ and the expected value of the output is at least $OPT - O(\epsilon n^2)$.*

Note that the use of Algorithm 2.1 in Algorithm 2.2 rather than an arbitrary approximation algorithm is crucial for our proof to work with a sample of size $O(1/\epsilon^4)$.

The third algorithm, which we have analyzed for max cut only, eliminates the preliminary random sample phase altogether; it is just the naïve greedy algorithm with random order, but it is repeated enough times to guarantee near-optimality. It is slowest but simplest in design and does not even really require prior knowledge of ϵ: one could just run it until satisfied with the quality of the current cut, with the reassuring knowledge that it will converge to a near-optimal value in polynomial time.

Theorem 2.3. *For any $\epsilon > 0$, the expected value of the output of Algorithm 2.3 is at least $OPT - \epsilon n^2$.*

Interestingly one of the best greedy algorithms in our correlation clustering experiments reported

Repeat $2^{2^{\tilde{O}(poly(\epsilon))}}$ times
for each vertex v of V in random order **do**
  Place v on the side that maximizes the number of resulting crossing edges.
end for
Output the best cut found

Figure 2.3: Most naïve greedy algorithm, for MAX CUT only

in Chapter 6 is greedy with random order, i.e. one iteration of Algorithm 2.3. Unfortunately our results do not apply in this setting because the number of clusters was not limited to a constant.

Warning: the proofs in this chapter are in the process of being rewritten. Comprehensibility is not guaranteed.

2.2 Analyzing Algorithm ?? for max cut

2.2.1 Definitions and notations

The problem. Each vertex can take on 2 possible values $i \in \{1, 2\}$, corresponding to the two sides of the cut. The goal is to find an cut of the vertices, among the 2^n possible cuts, so as to maximize the number of crossing edges. We describe a cut by an $2n$-dimensional vector x, where $x_{ui} \in \{0, 1\}$ equals 1 if and only if the variable associated to vertex u has value equal to i. We can then write the objective function z that needs to be minimized as:

$$z(x) = \sum_{\substack{1 \le u_1, u_2 \le n, \\ 1 \le i_1, i_2 \le k}} a_{u_1,i_1,u_2,i_2} x_{u_1 i_1} x_{u_2 i_2}, \tag{2.1}$$

where $A = (a_{u_1,i_1,u_2,i_2})$ is an $2n$-dimensional array (a matrix), symmetric under permutation of the 2 indices (u_j, i_j)'s, such that $a_{u_1 i_1 u_2 i_2} = 1/2$ whenever $i_1 = i_2$ and $\{u_1, u_2\} \in E$ and 0 otherwise. It will also be useful to write $z(x) = A(x, x)$, using the bilinear function

$$A(x^{(1)}, x^{(2)}) = \sum_{\substack{1 \le u_1, u_2 \le n, \\ 1 \le i_1, i_2 \le 2}} a_{u_1,i_1,u_2,i_2} x^{(1)}_{u_1 i_1} x^{(2)}_{u_2 i_2}. \tag{2.2}$$

The algorithm. The analysis will focus on the run when the sample S is assigned in the same way as OPT. For the sake of analysis pretend that the sample is chosen one vertex at a time rather than batched. Let "time t" denote the instant after the first t vertices are considered by the algorithm. At any time $t \in [0, n]$, Algorithm **??** defines a partial cut x^t:

$$x^t_{ui} = \begin{cases} 1 & \text{if vertex } u \text{ has been given value } i \text{ by time } t \\ 0 & \text{if vertex } u \text{ has been given some value} \neq i \\ & \quad \text{by time } t \\ 0 & \text{if vertex } u \text{ has not yet been given a value} \\ & \quad \text{by time } t \end{cases}$$

Thus x^0 is uniformly equal to 0, and x^n describes the output of the algorithm. We find it convenient to define x_v^t to be a 2-dimensional vector with components x_{vi}^t, and similarly for the other $2n$-dimensional vectors.

Let r_t denote the t^{th} vertex considered by the algorithm. Let $x^* = (x_{ui}^*)$ be the optimal cut. In the case when $t \le t_0$, we are in the initial phase and the vertex $u = r_t$ considered by the algorithm at time t is placed according to the optimal cut, so that the 2-dimensional vector describing the placement of vertex u is $x_u^t = x_u^*$. In the case when $t > t_0$, the algorithm decides in a greedy fashion: let $b(x)$ be the $2n$-dimensional vector of partial derivatives of $z(x)$; since z is bilinear and $A(x^t - x^{t-1},\ x^t - x^{t-1}) = 0$, $b_{ui}(x)$ is the increase of $z(x)$ when we increase x_{ui} by 1. Then, by definition of greedy, $x_{r_t}^t = x_{r_t}^{t-1} + g_{r_t}^t$, where

$$g_{vi}^t = \begin{cases} x_{vi}^* & \text{if } t \le t_0 \\ 1 & \text{if } t > t_0 \text{ and } i = \arg\min_j b_{vj}(x^{t-1}) \\ 0 & \text{otherwise} \end{cases}$$

Our analysis computes x_u^t and $b_u(x^t)$ inductively.

The fictitious cut. Instead of analyzing x^t directly, the analysis will instead focus on the following auxiliary variables. Let $S^t = \{r_1, r_2 \cdots r_t\}$ be the vertices placed up to time t. We extrapolate the partial cut x^t into a vector $\hat{x}^t$ that is a complete (fractional) cut by examining, for each $v \notin S^t$, how v would have been placed if it had been placed at time $\tau \le t$, and taking the average over all past times τ:

$$\hat{x}_v^t = \begin{cases} x_v^t & \text{if } v \in S^t \\ (1/t)\sum_{\tau=1}^t g_v^\tau & \text{if } v \notin S^t. \end{cases}$$

2.2.2 Proof of Theorem ?? for max cut

In the first lemma, we bound the decrease in the value of the fictitious cut when going from time $t-1$ to time t, and the bound uses the derivative of the objective function.

Lemma 2.4. *For every t, we have $z(\hat{x}^t) - z(\hat{x}^{t-1}) \ge (\hat{x}^t - \hat{x}^{t-1}) \cdot b(\hat{x}^{t-1}) - 4n^2/t^2$.*

Proof. Note that $b(x)_{u_1 i_1} = 2\sum_{u_2,i_2} a_{u_1 i_1 u_2 i_2} x_{u_2 i_2}$, so that $A(x^{(1)},\ x^{(2)}) = (1/2)x^{(1)} \cdot b(x^{(2)})$. Let $\hat{s}^t = \hat{x}^t - \hat{x}^{t-1}$. By multilinearity and symmetry of A,

$$\begin{aligned} z(\hat{x}^t) &= A(\hat{x}^{t-1} + \hat{s}^t,\ \hat{x}^{t-1} + \hat{s}^t) \\ &= A(\hat{x}^{t-1},\ \hat{x}^{t-1}) + \hat{s}^t \cdot b(\hat{x}^{t-1}) + A(\hat{s}^t,\ \hat{s}^t). \end{aligned}$$

A short calculation proves the following expression.

$$\hat{s}_u^t = \left(g_u^t - \hat{x}_u^{t-1}\right) \begin{cases} 1 & \text{if } u = r_t \text{ (is being placed at time } t) \\ (1/t) & \text{if } u \notin S^t \text{ (has not yet been placed)} \\ 0 & \text{if } u \in S^{t-1} \text{ (has already been placed)} \end{cases} \tag{2.3}$$

Since both g_u^t and $\hat{x}_u^{t-1}$ have ℓ_1 norm equal to 1, we obtain that the ℓ_1-norm of $\hat{s}^t$ is

$$\sum_u |\hat{s}_u^t| \leq 2 + \frac{2}{t}(n-t) = 2\frac{n}{t}. \tag{2.4}$$

Then

$$|A(\hat{s}^t, \hat{s}^t)|_1 \leq |A|_\infty |\hat{s}^t|_1^2 \leq 4\frac{n^2}{t^2}.$$

Replacing $A(\hat{x}^{t-1}, \hat{x}^{t-1})$ by $z(\hat{x}^{t-1})$ concludes the proof. □

In the second lemma, we bound the expectation of the decrease, and relate it to the difference between the values of the fictitious cut and of the (scaled) true cut, when evaluated by the derivative function b.

Lemma 2.5. *For every t, the expected value of $z(\hat{x}^t) - z(\hat{x}^{t-1})$ is greater than or equal to*

$$-4n^2/t^2 - 2\frac{n}{t(n-t+1)}\mathbf{E}\left[|b(\hat{x}^{t-1}) - b(\frac{n}{t}x^{t-1})|_1\right].$$

Proof. First apply Lemma 2.4. Then, write:

$$\begin{aligned}
&\hat{s}^t \cdot b(\hat{x}^{t-1}) \\
&= \hat{s}^t \cdot b(\frac{n}{t}x^{t-1}) + \hat{s}^t \cdot (b(\hat{x}^{t-1}) - b(\frac{n}{t}x^{t-1})) \\
&= (\frac{n}{t})\hat{s}^t \cdot b(x^{t-1}) + \hat{s}^t \cdot (b(\hat{x}^{t-1}) - b(\frac{n}{t}x^{t-1})) \\
&\geq \hat{s}^t \cdot (b(\hat{x}^{t-1}) - b(\frac{n}{t}x^{t-1})),
\end{aligned}$$

where the equality follows from the fact that the function $b(y)$ is linear. To justify the last inequality, recall that the greedy choice g_u^t maximizes $y \cdot b_u(x^{t-1})$ for y with $|y|_1 = 1$ and $y_{ui} \in [0,1]$. Going back to (2.3), we see that $\hat{s}^t \cdot b(x^{t-1})$ is greater than or equal to 0. This is where the greedy definition of the algorithm comes into the analysis.

Given the history S^{t-1}, we can write:

$$\begin{aligned}
&\mathbf{E}\left[\hat{s}^t \cdot (b(\hat{x}^{t-1}) - b(\frac{n}{t}x^{t-1}))|S^{t-1}\right] \\
&= \sum_v \mathbf{E}\left[\hat{s}_v^t \cdot (b_v(\hat{x}^{t-1}) - b_v(\frac{n}{t}x^{t-1}))|S^{t-1}\right] \\
&= \sum_v \mathbf{E}\left[\hat{s}_v^t|S^{t-1}\right] \cdot (b_v(\hat{x}^{t-1}) - b_v(\frac{n}{t}x^{t-1})) \\
&\geq -\sum_v |\mathbf{E}\left[\hat{s}_v^t|S^{t-1}\right]|_1 \cdot |b_v(\hat{x}^{t-1}) - b_v(\frac{n}{t}x^{t-1})|_1 \\
&\geq -2\frac{n}{t(n-t+1)}|b(\hat{x}^{t-1}) - b(\frac{n}{t}x^{t-1})|_1,
\end{aligned}$$

where the second equality follows from the fact that given S^{t-1}, $b(\hat{x}^{t-1})$ is fixed, the next inequality follows from general principles. As for the last inequality, fix S^{t-1} and a vertex u and consider evaluating $\mathbf{E}\left[\hat{s}_u^t|S^{t-1}\right]$. If $u \in S^{t-1}$, then $\hat{s}_u^t = 0$ and so the expectation is zero. Otherwise, by the random order assumption and (2.4),

$$\mathbf{E}\left[|\hat{s}_u^t|_1|S^{t-1}\right] = \mathbf{E}\left[\frac{|\hat{s}^t|_1}{n-t+1}|S^{t-1}\right] \leq 2\frac{n}{t(n-t+1)}$$

and the last inequality follows. This concludes the proof of Lemma 2.5. □

The next lemma bounds the difference between the b-values of the fictitious cut and of the (scaled) true cut.

Lemma 2.6. *For every t, we have $\mathbf{E}\left[|b(\hat{x}^t) - b(\frac{n}{t}x^t)|_1\right] = O(\sigma)$, where $\sigma = O\left(\frac{n^2}{\sqrt{t}}\sqrt{\frac{n-t}{n}}\right)$.*

The proof is based on a certain martingale property and deferred to the next section.

The next lemma relates the z-values of the fictitious cut at two different times.

Lemma 2.7. *For every $t \geq t_0$ we have $\mathbf{E}\left[z(\hat{x}^t)\right] - \mathbf{E}\left[z(\hat{x}^{t_0})\right] \geq -O(\epsilon n^2)$.*

Proof. Use Lemma 2.5 for every τ from t_0 to t, and sum; apply Lemma 2.6 to each term in the sum, and use $t_0 \geq 1/\epsilon^2$: after a short calculation we get:

$$\mathbf{E}\left[z(x^t) - z(\hat{x}^{t_0})\right] = -n^2 O\left(\frac{1}{t_0} + \frac{1}{\sqrt{t_0}} + \frac{1}{\sqrt{t}}\right) = -O(\epsilon n^2).$$

□

Proof. (of Theorem **??**.) By definition of the fictitious cut, the output cut x^n is equal to $\hat{x}^n$. Apply Lemma 2.7 for $t = n$; and recall that by definition of x^*, $\hat{x}^{t_0}$ is the optimal cut. □

2.2.3 Proof of Lemma 2.6 using martingales.

In this section, we detail the proof of Lemma 2.6. Fix v. The following lemma is simple, yet central. This martingale is the key to all the future applications of the Azuma-Hoeffding inequality.

Lemma 2.8. *$Z_v^t = \frac{t}{n-t}(\hat{x}_v^t - (n/t)x_v^t)$ is a martingale.*

Proof. If $v \in S^{t-1}$ then $\hat{x}_v^t = x_v^t = \hat{x}_v^{t-1} = x_v^{t-1}$. Let x be their common value. Then

$$\frac{t}{n-t}(\hat{x}_v^t - (n/t)x_v^t) = \frac{t-1}{n-t+1}(\hat{x}_v^{t-1} - (n/(t-1))x_v^{t-1})$$

are both equal to $-x$ and the martingale statement holds in that case. If $v \notin S^{t-1}$, then $x_v^{t-1} = 0$; by the random order, with probability $1/(n-t+1)$ we have $v = r_t$ and $x_v^t = \hat{x}_v^t = g_v^t$; with the remaining probability $(n-t)/(n-t+1$, we have $x_v^t = 0$ and $\hat{x}_v^t = ((t-1)/t)\hat{x}_v^{t-1} + (1/t)g_v^t$, and so:

$$\begin{aligned}
\mathbf{E}\left[\frac{t}{n-t}(\hat{x}_v^t - (n/t)x_v^t)|S^{t-1}\right] = & \\
& \frac{1}{n-t+1}(-g_v^t) + \\
& \frac{n-t}{n-t+1} \cdot \frac{t}{n-t}\left[\frac{t-1}{t}\hat{x}_v^{t-1} + \frac{1}{t}g_v^t\right] \\
= & \quad \frac{t-1}{n-t+1}\hat{x}_v^{t-1},
\end{aligned}$$

which concludes the proof since $x_v^{t-1} = 0$ in that case. □

Lemma 2.9. *$B^t_{vi} = \frac{t}{n-t}(b_{vi}(\hat{x}^t) - b_{vi}(\frac{n}{t}x^t))$ is a martingale, with stepsize bounded by $4n/(n-t)$.*

Proof. The martingale statement follows from Lemma 2.8 by linearity: $b_{vi}(x) = \sum_{uj} \alpha_{uj} x_{uj}$. To bound its step-size, we note that $\alpha = \max |\alpha_{uj}| \leq 2$ (since $|A|_\infty \leq 1$), so that the step size is at most:

$$\left| \sum_{uj} \alpha_{uj} Z^t_{uj} - \sum_{uj} \alpha_{uj} Z^{t-1}_{uj} \right| \leq \alpha \sum_u |Z^t_u - Z^{t-1}_u|_1.$$

When $u \in S^{t-1}$, we have $|Z^t_u - Z^{t-1}_u|_1 = 0$. When $u = r_t$, we have

$$|Z^t_u - Z^{t-1}_u|_1 = \left| \frac{-g^t_v}{n-t+1} - \frac{(t-1)\hat{x}^{t-1}_v}{n-t+1} \right|_1 \leq \frac{t}{n-t+1}.$$

When $u \notin S^t$, we have

$$\begin{aligned} |Z^t_u - Z^{t-1}_u|_1 &= \left| \frac{(t-1)\hat{x}^{t-1}_u + g^t_u}{n-t} - \frac{t-1}{n-t+1}\hat{x}^{t-1}_u \right|_1 \\ &\leq \frac{t-1}{(n-t)(n-t+1)} + \frac{1}{n-t} \\ &= \frac{n}{(n-t)(n-t+1)}. \end{aligned}$$

Summing, the stepsize is at most $\alpha((t+n)/(n-t+1)) \leq 4n/(n-t)$. □

We recall the Azuma-Hoeffding inequality.

Theorem 2.10. *[Azuma-Hoeffding] Let $X_0, X_1, \ldots, X_t$ be a martingale such that $|X_k - X_{k-1}| \leq c_k$ for all k. Then, for all $\lambda > 0$,*

$$\Pr(|X_t - X_0| \geq \lambda) \leq 2e^{-\lambda^2/(2\sum_{k=1}^t c_k^2)}.$$

Proof. (of Lemma 2.6) From Lemma 2.9, $|b(\hat{x}^t) - b(\frac{n}{t}x^t)|_1$ is also a martingale with stepsize at most $4n/(n-t)$. Apply the Azuma-Hoeffding inequality:

$$\Pr\left(\left| b(\hat{x}^t) - b(\frac{n}{t}x^t) \right|_1 \geq \lambda \right) \leq e^{-\lambda^2/\sigma^2} \tag{2.5}$$

where $\sigma = O\left(\frac{n^2}{\sqrt{t}}\sqrt{\frac{n-t}{n}}\right)$. Finally, integrate over λ, using the fact that for a non-negative random variable X, $\mathbf{E}[X] = \int_{\lambda=0}^{\infty} \Pr(X \geq \lambda)\, d\lambda$. □

2.3 Analyzing Algorithm ??

2.3.1 Definitions and Notations

The input to the max-rCSP problem is a set of constraints, where each constraint is over a collection of r variables x_v, a subset of the n possible variables: $1 \leq v \leq n$. Each variable can take on k possible values $i \in [1, k]$, and the constraint specifies which of the k^r assignments satisfy the constraint. The goal is to find an assignment of the variables, among the k^n possible assignments, so as to

maximize the number of satisfied constraints. MAX CUT vertices and edges correspond to MAX CSP variables and constraints respectively. We will actually view the problem as *minimizing* the number of unsatisfied constraints.

We describe an assignment of the variables by an nk-dimensional vector x, where $x_{ui} \in \{0,1\}$ equals 1 if and only if the variable associated to variable u has value equal to i. We can then write the objective function z that needs to be maximized as:

$$z(x) = \sum_{\substack{1\le u_1,\cdots,u_r\le n,\\ 1\le i_1,\ldots,i_r\le k}} a_{u_1,i_1,u_2,i_2,\ldots,u_r,i_r} x_{u_1 i_1} x_{u_2 i_2} \cdots x_{u_r i_r}, \tag{2.6}$$

where $A = (a_{u_1,i_1,u_2,i_2,\ldots,u_r,i_r})$ is an nk-dimensional array, symmetric under permutation of the r indices (u_j, i_j)'s. For example for max cut $a_{u,i,v,j} = \frac{1}{2}\mathbb{1}\,(i \neq j$ and $\exists$ edge between u and $v)$. As before we write $z(x) = A(x, x, \cdots, x)$ using the multilinear function

$$A(x^{(1)}, \cdots, x^{(r)}) \quad = \sum_{\substack{1\le u_1,,\cdots,u_r\le n,\\ 1\le i_1,\ldots,i_r\le k}} a_{u_1,i_1,u_2,i_2,\ldots,u_r,i_r} x^{(1)}_{u_1 i_1} x^{(2)}_{u_2 i_2} \cdots x^{(r)}_{u_r i_r}. \tag{2.7}$$

We define x^t_{ui}, r_t, and g as in the max cut case. Let $b(x)$ be the kn-dimensional vector of partial derivatives of $z(x)$; since z is multilinear and $A(x^t - x^{t-1}, \ldots,\ x^t - x^{t-1}) = 0$, $b_{ui}(x)$ is the increase of $z(x)$ when we increase x_{ui} by 1. Observe that

$$b_{ui}(x) = r \cdot A(e_{ui}, x, \ldots, x) \tag{2.8}$$

where e_{ui} is a unit vector.

2.3.2 Proof of Theorem ??

Lemma 2.11.

$$|A(x^{(1)}, \cdots, x^{(r)})| \le |A|_\infty |x^{(1)}|_1 |x^{(2)}|_1 \cdots |x^{(r)}|_1.$$

Proof. Simple and omitted. □

Lemma 2.12. *If $|\delta x|_1 \le |x|_1$, then*

$$|A(x+\delta x, \cdots, x+\delta x) - A(x, \cdots, x) - rA(\delta x, x, \cdots, x)| \le 2^r |A|_\infty |\delta x|^2 |x|_1^{r-2}.$$

Proof. By symmetry and multilinearity we have

$$A(x+\delta x, \cdots, x+\delta x) = \sum_i \binom{r}{i} A(\underbrace{\delta x, \cdots, \delta x}_{i}, x, \cdots, x).$$

Using Lemma 2.11 for every $i \ge 2$, the assumption $|\delta x|_1 \le |x|_1$, and the fact that $\sum_{i\ge 2}\binom{r}{i} \le 2^r$, yields the lemma. □

Lemma 2.13 (Analog of Lemma 2.4)**.** *For every t, we have $z(\hat{x}^t) - z(\hat{x}^{t-1}) \ge (\hat{x}^t - \hat{x}^{t-1}) \cdot b(\hat{x}^{t-1}) - 2^{r+2} n^r / t^r$.*

Proof. Let $\hat{s}^t = \hat{x}^t - \hat{x}^{t-1}$. We use Lemma 2.12 to write:

$$\begin{aligned} z(\hat{x}^t) - z(\hat{x}^{t-1}) &= A(\hat{x}^{t-1} + \hat{s}^t, \cdots, \hat{x}^{t-1} + \hat{s}^t) - A(\hat{x}^{t-1}, \cdots, \hat{x}^{t-1}) \\ &= rA(\hat{s}^t, \hat{x}^{t-1}, \cdots, \hat{x}^{t-1}) + 2^r |A|_\infty |\hat{s}^t|_1^2 |\hat{x}^{t-1}|_1^{r-2}. \end{aligned}$$

By linearity and (2.8), we have $rA(\hat{s}^t, \hat{x}^{t-1}, \cdots, \hat{x}^{t-1}) = \hat{s}^t \cdot b(\hat{x}^{t-1})$, which gives the first term in the right hand side of Lemma 2.13. By definition of max-r-CSP, we have $|A|_\infty \leq 1$. Since $\hat{x}^{t-1}$ is a complete fractional assignment of values to the variables, $|\hat{x}^{t-1}|_1$ is equal to n, the number of variables. Finally, it is not difficult to check that $|\hat{s}^t|_1 \leq (1 + n/t) \leq 2n/t$. Indeed a short calculation proves the following expression.

$$\hat{s}_u^t = \left(g_u^t - \hat{x}_u^{t-1}\right) \begin{cases} 1 & \text{if } u = r_t \text{ (is being placed at time } t) \\ (1/t) & \text{if } u \notin S^t \text{ (has not yet been placed)} \\ 0 & \text{if } u \in S^{t-1} \text{ (has already been placed)} \end{cases} \tag{2.9}$$

Since both g_u^t and $\hat{x}_u^{t-1}$ have ℓ_1 norm equal to 1, we obtain that the ℓ_1-norm of $\hat{s}^t$ is

$$\sum_u |\hat{s}_u^t| \leq 2 + \frac{2}{t}(n - t) = 2\frac{n}{t}. \tag{2.10}$$

Replacing $A(\hat{x}^{t-1}, \hat{x}^{t-1})$ by $z(\hat{x}^{t-1})$ concludes the proof. □

In the second lemma, we bound the expectation of the decrease, and relate it to the difference between the values of the fictitious cut and of the (scaled) true cut, when evaluated by the derivative function b.

Lemma 2.14 (Analog of Lemma 2.5)**.** *For every t, the expected value of $z(\hat{x}^t) - z(\hat{x}^{t-1})$ is greater than or equal to*

$$-2^{r+2} n^r / t^r - 2\frac{n}{t(n-t+1)} \mathbf{E}\left[|b(\hat{x}^{t-1}) - b(\frac{n}{t} x^{t-1})|_1 \right].$$

Proof. First apply Lemma 2.13. Then, write:

$$\begin{aligned} &\hat{s}^t \cdot b(\hat{x}^{t-1}) \\ &= \hat{s}^t \cdot b(\frac{n}{t} x^{t-1}) + \hat{s}^t \cdot (b(\hat{x}^{t-1}) - b(\frac{n}{t} x^{t-1})) \\ &= (\frac{n}{t})^{r-1} \hat{s}^t \cdot b(x^{t-1}) + \hat{s}^t \cdot (b(\hat{x}^{t-1}) - b(\frac{n}{t} x^{t-1})) \\ &\geq \hat{s}^t \cdot (b(\hat{x}^{t-1}) - b(\frac{n}{t} x^{t-1})), \end{aligned}$$

where the second equality follows from the fact that the function $b(y)$ is $(r-1)$-multilinear. To justify the last inequality, recall that the greedy choice g_u^t maximizes $y \cdot b_u(x^{t-1})$ for y with $|y|_1 = 1$ and $y_{ui} \in [0, 1]$. Going back to (2.9), we see that $\hat{s}^t \cdot b(x^{t-1})$ is greater than or equal to 0. This is where the greedy definition of the algorithm comes into the analysis.

Given the history S^{t-1}, we can write:

$$\begin{aligned}
&\mathbf{E}\left[\hat{s}^t \cdot (b(\hat{x}^{t-1}) - b(\frac{n}{t}x^{t-1}))|S^{t-1}\right] \\
&= \sum_v \mathbf{E}\left[\hat{s}^t_v \cdot (b_v(\hat{x}^{t-1}) - b_v(\frac{n}{t}x^{t-1}))|S^{t-1}\right] \\
&= \sum_v \mathbf{E}\left[\hat{s}^t_v|S^{t-1}\right] \cdot (b_v(\hat{x}^{t-1}) - b_v(\frac{n}{t}x^{t-1})) \\
&\geq -\sum_v |\mathbf{E}\left[\hat{s}^t_v|S^{t-1}\right]|_1 \cdot |b_v(\hat{x}^{t-1}) - b_v(\frac{n}{t}x^{t-1})|_1 \\
&\geq -2\frac{n}{t(n-t+1)}|b(\hat{x}^{t-1}) - b(\frac{n}{t}x^{t-1})|_1,
\end{aligned}$$

where the second equality follows from the fact that given S^{t-1}, $b(\hat{x}^{t-1})$ is fixed, the next inequality follows from general principles. As for the last inequality, fix S^{t-1} and a vertex u and consider evaluating $\mathbf{E}\left[\hat{s}^t_u|S^{t-1}\right]$. If $u \in S^{t-1}$, then $\hat{s}^t_u = 0$ and so the expectation is zero. Otherwise, by the random order assumption and (2.10),

$$\mathbf{E}\left[|\hat{s}^t_u|_1|S^{t-1}\right] = \mathbf{E}\left[\frac{|\hat{s}^t|_1}{n-t+1}|S^{t-1}\right] \leq 2\frac{n}{t(n-t+1)}$$

and the last inequality follows. This concludes the proof of Lemma 2.14. □

The next lemma bounds the difference between the b-values of the fictitious cut and of the (scaled) true cut.

Lemma 2.15 (Analog of Lemma 2.6)**.** *For every t, we have $\mathbf{E}\left[|b(\hat{x}^t) - b(\frac{n}{t}x^t)|_1\right] = O(\sigma)$, where $\sigma = O\left(\frac{n^r}{\sqrt{t}}\sqrt{\frac{n-t}{n}}\right)$.*

The proof is based on a certain martingale property and deferred to the next section.

The next lemma relates the z-values of the fictitious assignment at two different times.

Lemma 2.16 (Analog of Lemma 2.7)**.** *For every $t \geq t_0$ we have $\mathbf{E}\left[z(\hat{x}^t)\right] - \mathbf{E}\left[z(\hat{x}^{t_0})\right] \geq -O(\epsilon n^r)$.*

Proof. Use Lemma 2.14 for every τ from t_0 to t, and sum; apply Lemma 2.15 to each term in the sum, and use $t_0 \geq 1/\epsilon^2$: after a short calculation we get:

$$\mathbf{E}\left[z(x^t) - z(\hat{x}^{t_0})\right] = -n^r O\left(\frac{1}{t_0} + \frac{1}{\sqrt{t_0}} + \frac{1}{\sqrt{t}}\right) = -O(\epsilon n^r).$$

□

Proof. (of Theorem **??**.) By definition of the fictitious assignment, the output assignment x^n is equal to $\hat{x}^n$. Apply Lemma 2.16 for $t = n$; and recall that by definition of x^*, $\hat{x}^{t_0}$ is the optimal assignment. □

2.3.3 Proof of Lemma 2.15 using martingales.

Let $C(\sigma)$ denote the set of random variables X such that for all $p > 0$ we have $\mathbf{E}\left[e^{p(X-\sigma)}\right] \leq e^{p^2\sigma^2}$. Intuitively $C(\sigma)$ is the set of random variables that are at most σ except for Chernoff-like tail probabilities. On the first reading one may read $X \in C(\sigma)$ as $\mathbf{E}[X] = O(\sigma)$.

This definition satisfies a number of easily verified properties.

Lemma 2.17. *Let σ and α be positive constants and X and Y be random variables. Then:*

- *If $X \in C(\sigma)$ then $\Pr(X - \sigma \geq \alpha) \leq e^{-\alpha^2/(4\sigma^2)}$.*
- *$\mathbf{E}[X] = O(\sigma)$.*
- *If $X \in C(\sigma)$ then $\alpha X \in C(\alpha\sigma)$.*
- *If $X \in C(\sigma)$ and $Y \leq X$ then $Y \in C(\sigma)$.*
- *The random variable with constant value σ is in $C(\sigma)$.*

Proof. The first is proved using the same as Chernoff-bounds:

$$\begin{aligned}\Pr(X - \sigma \geq \alpha) = \Pr\left(e^{p(X-\sigma)} \geq e^{p\alpha}\right) &\leq \mathbf{E}\left[e^{p(X-\sigma)}\right]/e^{p\alpha} \\ &\leq e^{p^2\sigma^2 - p\alpha} = e^{-\alpha^2/(4\sigma^2)}\end{aligned}$$

choosing $p = \alpha/(2\sigma^2)$ in the last equality.

The second follows easily from Jensen's inequality:[1]

$$e^1 = e^{\sigma^2/\sigma^2} \geq \mathbf{E}\left[e^{(X-\sigma)/\sigma}\right] \geq e^{(\mathbf{E}[X]-\sigma)/sigma}$$

hence $\mathbf{E}[X]/\sigma \leq 2$. The remainder are straightforward. □

Furthermore this class of random variables adds nicely:

Lemma 2.18. *For any real-valued random variables X, Y, if $X \in C(\sigma_x)$ and $Y \in C(\sigma_y)$, then $X + Y \in C(\sigma_x + \sigma_y)$.*

The intuition for the proof is that the worst case is when X and Y are proportional, in which case the Lemma is trivial.

Proof. Let $p = (\sigma_x + \sigma_y)/\sigma_x$ and $q = (\sigma_x + \sigma_y)/\sigma_y$, which are Hölder conjugates, i.e. $1/p + 1/q = 1$. For any $\alpha > 0$ we have:

$$\begin{aligned}\mathbf{E}\left[e^{\alpha(X+Y-\sigma_X-\sigma_Y)}\right] = \mathbf{E}\left[e^{\alpha(X-\sigma_X)} \cdot e^{\alpha(Y-\sigma_Y)}\right] &\leq \mathbf{E}\left[e^{p\alpha(X-\sigma_X)}\right]^{1/p} \mathbf{E}\left[e^{q\alpha(X-\sigma_X)}\right]^{1/q} \\ &\leq e^{(p^2\alpha^2\sigma_X^2)/p + (q^2\alpha^2\sigma_Y^2)/q} = e^{\alpha^2(\sigma_x+\sigma_y)\sigma_x + \alpha^2(\sigma_x+\sigma_y)\sigma_y} \\ &= e^{\alpha^2(\sigma_x+\sigma_y)^2}\end{aligned}$$

as desired. □

[1] Alternatively, it follows from integrating the first.

Theorem 2.19. *A martingale with sum of squared step-size bounds σ^2 is in $C(\sigma/2)$.*

Proof. From the standard proof of the Azuma-Hoeffding inequality we have $\mathbf{E}\left[e^{pX}\right] \leq e^{p^2\sigma^2/2}$ hence $\mathbf{E}\left[e^{p(X-\sigma)}\right] \leq e^{p^2\sigma^2/2}$. □

We will actually prove the following Lemma, which trivially implies Lemma 2.15

Lemma 2.20. *For every t and q-dimensional array A with $|A|_\infty \leq 1$ we have*

$$\Pr\left(|A(\frac{n}{t}x^t, \frac{n}{t}x^t \cdots \frac{n}{t}x^t) - A(\hat{x}^t, \cdots \hat{x}^t)| \geq \sigma + \lambda\right) \leq e^{-\lambda^2/\sigma^2}$$

where $\sigma = O(n^q\sqrt{\frac{n-t}{nt}})$.

Proof of Lemma 2.20. To show this we prove by induction on r that for an r-dimensional symmetric array with $|A|_\infty \leq 1$,

$$|A(x^t, x^t \cdots x^t) - A(\bar{x}^t, \cdots \bar{x}^t)| \in C(O(t^r\sqrt{\frac{n-t}{nt}})).$$

The base case $r = 1$ was already done in Section 2.2.3. Without loss of generality suppose that A is symmetric. We have:

$$\begin{aligned}
&\frac{dA(x, x, \cdots x)/(n-t)^r}{dt} \\
&= \frac{dA/dt}{(n-t)^r} + \frac{rA}{(n-t)^{r+1}} \\
&= \frac{1}{(n-t)^r}\Bigg[rA(dx/dt, x \cdots x) \\
&\qquad + rA(x, x, \cdots x)/(n-t)\Bigg] \\
&= \frac{r}{(n-t)^r}A(dx/dt + \frac{x}{n-t}, x \cdots x).
\end{aligned}$$

Therefore, using Lemma 2.12

$$A(x^t, \cdots x^t)/(n-t)^r - A(x^{t-1}, \cdots x^{t-1})/(n-(t-1))^r =$$

$$\frac{r}{(n-t)^r}\left[A(x^t - x^{t-1} + \frac{x^{t-1}}{n-t}, x^{t-1} \cdots x^{t-1}) + O(t^{r-2})\right]$$

Define:

$$D^t \equiv A(\Delta x + x/(n-t), x, x) - A(\Delta\bar{x} + \bar{x}/(n-t), \bar{x}, \bar{x})$$

We have:

$$\begin{aligned}
&\frac{A(x^\tau) - A(\bar{x}^\tau)}{(n-\tau)^r} \\
&= \frac{A(x^0) - A(\bar{x}^0)}{(n-0)^r} + \sum_{t=1}^{\tau} \frac{r}{(n-t)^r}(D^t + O(t^{r-2})) \\
&= 0 + \sum_{t=1}^{\tau} \frac{r}{(n-t)^r}(D^t - \mathbf{E}\left[D^t | hist\right]) \\
&\quad + \sum_{t=1}^{\tau} \frac{r}{(n-t)^r}\mathbf{E}\left[D^t | hist\right]) \\
&\quad + \sum_{t=1}^{\tau} \frac{r}{(n-t)^r} O(t^{r-2}) \qquad (2.11)
\end{aligned}$$

Therefore:

$$\begin{aligned}
&\left|\frac{A(x^\tau) - A(\bar{x}^\tau)}{(n-\tau)^r}\right| \\
&\leq \left|\sum_{t=1}^{\tau} \frac{r}{(n-t)^r}(D^t - \mathbf{E}\left[D^t | hist\right])\right| \\
&\quad + \sum_{t=1}^{\tau} \frac{r}{(n-t)^r}\left|\mathbf{E}\left[D^t | hist\right])\right| \\
&\quad + \sum_{t=1}^{\tau} \frac{r}{(n-t)^r} O(t^{r-2}) \qquad (2.12)
\end{aligned}$$

By Lemma 2.18, it is sufficient to analyze each of these terms separately. First consider the first term of Equation 2.12. The partial sums form a martingale. Clearly the uncertain part of D, $|A(\Delta x, x, x) - A(\Delta\bar{x}, \bar{x}, \bar{x})|$ is bounded by t^{r-1}, so the step size is at most $O(\frac{t^{r-1}r}{(n-t)^r})$. We have:

$$\begin{aligned}
\sigma^2 &= O(1)\sum_{t=1}^{\tau}(\frac{t^{r-1}}{(n-t)^r})^2 \\
&\leq \begin{cases} O(1)\sum_{t=1}^{\tau}(\frac{t^{r-1}}{(n)^r})^2 & \text{if } \tau \leq n/2 \\ O(1)\sum_{t=1}^{\tau}(\frac{n^{r-1}}{(n-t)^r})^2 & \text{if } \tau > n/2 \end{cases} \\
&= \begin{cases} O(1)\frac{\tau^{2r-1}}{(n)^{2r}} & \text{if } \tau \leq n/2 \\ O(1)\frac{n^{2r-2}}{(n-\tau)^{2r-1}} & \text{if } \tau > n/2 \end{cases} \\
&= \begin{cases} O(1)(\frac{\tau^{r-1/2}}{(n-\tau)^r})^2 & \text{if } \tau \leq n/2 \\ O(1)(\frac{\tau^{r-1}}{(n-\tau)^{r-1/2}})^2 & \text{if } \tau > n/2 \end{cases}
\end{aligned}$$

This is of the desired form except for the $\tau > n/2$ case, but liberal use of $\tau = \Theta(n)$ in that case remedies it:

$$\begin{aligned}
\frac{\tau^{r-1}}{(n-\tau)^{r-1/2}} &= \frac{\tau^{r-1/2}}{(n-\tau)^r}\sqrt{\frac{n-\tau}{\tau}} \\
&= \Theta(1)\frac{\tau^{r-1/2}}{(n-\tau)^r}\sqrt{\frac{n-\tau}{n}}
\end{aligned}$$

the Hoeffding-Azuma Lemma 2.19 finishes the proof that the first term of Equation 2.12 is small.

To analyze the second term of Equation 2.12 let $A(vi, x, x)$ denote the array A applied to a unit-vector with 1 in the vi component and xs. Note that $A(vi, x, x) = B_{vi}(x, x)$, where B_{vi} is an $r-1$ dimensional array.

$$\begin{aligned} D^t = & \\ & A(\Delta x + x/(n-t), x, x) \\ & - A(\Delta \bar{x} + \bar{x}/(n-t), \bar{x}, \bar{x}) \\ & + A(\Delta \bar{x} + \bar{x}/(n-t), x, x) \\ & - A(\Delta \bar{x} + \bar{x}/(n-t), x, x) \\ & = \sum_{vi} \Bigg[(\Delta x + x/(n-t) - \Delta \bar{x} - \bar{x}/(n-t))_{vi} A(vi, x, x) \\ & \qquad + (\Delta \bar{x} + \bar{x}/(n-t))_{vi} (A(vi, x, x) - A(vi, \bar{x}, \bar{x})) \Bigg] \end{aligned} \tag{2.13}$$

Recall that we are trying to bound $|\mathbf{E}[D^t | hist])|$. From Lemma 2.8 we know

$$\mathbf{E}[\Delta x + x/(n-t) - \Delta \bar{x} - \bar{x}/(n-t) | hist] = 0$$

disposing of the first term of Equation 2.13.

$$\begin{aligned} & \mathbf{E}[(\Delta \bar{x} + \bar{x}/(n-t))_{vi} (A(vi, x, x) - A(vi, \bar{x}, \bar{x})) | hist] \\ = \quad & \mathbf{E}[(\Delta \bar{x} + \bar{x}/(n-t)) | hist]_{vi} (A(vi, x, x) - A(vi, \bar{x}, \bar{x})) \end{aligned}$$

From the inductive hypothesis we know that $A(vi, x, x) - A(vi, \bar{x}, \bar{x})$ is probably small for any fixed v, i. Note that

$$\begin{aligned} |\mathbf{E}[\Delta \bar{x}_{vi} | hist]| \quad & \leq \quad \begin{cases} 1/(n-t) & \text{if not yet placed} \\ 1/n & \text{if already placed} \end{cases} \\ & \leq \quad 1/(n-t) \end{aligned}$$

and $|\bar{x}_{vi}|/(n-t) \leq t/(n(n-t)) \leq 1/(n-t)$. Therefore

$$\begin{aligned} & \Big| \mathbf{E}[(\Delta \bar{x} + \bar{x}/(n-t))_{vi} (A(vi, x, x) - A(vi, \bar{x}, \bar{x})) | hist] \Big| \\ & \leq \quad O(1)/(n-t) |A(vi, x, x) - A(vi, \bar{x}, \bar{x})|. \end{aligned}$$

Now by the inductive hypothesis:

$$|A(vi, x, x) - A(vi, \bar{x}, \bar{x})| \in C(O(t^{r-1} \sqrt{\frac{n-t}{nt}}))$$

We want to show:

$$(n-\tau)^r \sum_{t=1}^{\tau} \frac{r}{(n-t)^r} \left|\mathbf{E}\left[D^t | hist\right]\right| \in C(t^{r-1/2}\sqrt{(n-t)/t})$$

now using Lemmas 2.18 and 2.17 and the inductive hypothesis it remains to show:

$$(n-\tau)^r \sum_{t=1}^{\tau} \frac{r}{(n-t)^r} O(t^{r-1}\sqrt{\frac{n-t}{nt}}) \leq O(t^{r-1/2}\sqrt{(n-t)/t})$$

Splitting the sums into cases on $t < n/2$ and liberal use of r considered constant shows this sum is $O(1)t^{r-1/2}\sqrt{\frac{n-t}{n}}/(n-t)^r$

Summation shows that the third term of Equation 2.12 is also small. □

2.4 Analyzing Algorithm 2.2

Instead of Algorithm 2.2, we will analyze a variant, Algorithm 2.4.

Lemma 2.21. *With probability at least $1-\epsilon$, Algorithm 2.4 and Algorithm 2.2 give the same output.*

Proof. Consider a coupon collection problem with $N = 2^{t_0}$ coupons, one per cut of the first t_0 variables. Let $\kappa = N\ln(N/\epsilon) = 2^{t_0}(t_0 \ln 2 + \ln(1/\epsilon)) = 2^{O(t_0)}$ be the number of trials, one per cut in Y (excluding x^*). As long as Algorithm 2.4 collects all the coupons, it iterates over the same cuts of T as Algorithm 2.2 and hence returns the same result. Each coupon has probability $(1-1/N)^\kappa \leq e^{-\kappa/N} = \frac{\epsilon}{N}$ of not being collected, so a union bound shows that all are collected with probability at least $1-\epsilon$. □

In the analysis of the previous section, the construction always started by a cut induced on the sample by the optimal cut x^*. Now we need a new notation: let $x^t_{(y)}$ denote the construction at time t, starting from the cut induced on the sample by the cut y (instead of by the optimal cut x^*). Let $\hat{x}^t_{(y)}$ denote the fictitious cut at time t, starting from cut y. Formally, replace x^* in the definition of g^t_{vi} with y. All of our previous Lemmas still hold in this more general setting.

Lemma 2.22.

$$\mathbf{E}\left[\max_{y\in Y}\left|b(\frac{n}{t}x^t_{(y)}) - b(\hat{x}^t_{(y)})\right|_1\right] = \frac{1}{\epsilon}O(\sigma)$$

where $\sigma = O\left(\frac{n^{r-1}}{\sqrt{t}}\sqrt{\frac{n-t}{n}}\right)$.

Proof. Using Lemma 2.20 we have for every $y \in Y$:

$$\Pr\left(\left|b(\hat{x}^t_{(y)}) - b(\frac{n}{t}x^t_{(y)})\right|_1 \geq \lambda\right) \leq e^{-\lambda^2/\sigma^2}. \tag{2.14}$$

The set Y has size $2^{O(t_0)} = e^{O(1/\epsilon^2)}$, so by a union bound,

$$\Pr\left(\max_{y\in Y}\left|b(\hat{x}^t_{(y)}) - b(\frac{n}{t}x^t_{(y)})\right|_1 \geq \lambda\right) \leq e^{O(1/\epsilon^2)-\lambda^2/\sigma^2}$$

Finally, integrate $\min(1, e^{O(1/\epsilon^2)-\lambda^2/\sigma^2})$ over λ. □

- Let $t_1 = O(1/\epsilon^4)$ and $t_0 = O(1/\epsilon^2)$
- Let Y be a set of $2^{O(t_0)}$ random assignments to the n variables, plus the optimal assignment x^*.
- Choose a random permutation of the variables, the $\{r_t\}_{t=1}^n$, used in all future computations.
- Choose seed assignment $w = \arg\max_{y \in Y} z(x_{(y)}^{t_1})$
- Return assignment $x_{(w)}^n$

Figure 2.4: Variant algorithm for analysis

Lemma 2.23.

$$\mathbf{E}\left[\max_{y \in Y} |z(\frac{n}{t}x_{(y)}^t) - z(\hat{x}_{(y)}^t)|\right] = \frac{1}{\epsilon}O(\sigma)$$

where $\sigma = O\left(\frac{n^r}{\sqrt{t}}\sqrt{\frac{n-t}{n}}\right)$.

Proof. Identical to the proof of the previous Lemma, Lemma 2.22. □

2.4.1 Proof of Theorem 2.2

By definition of the fictitious cut, the output cut $x_{(w)}^n$ is equal to $\hat{x}_{(w)}^n$. As in the previous section, looking at the process from time t_1 to time n we write:

$$z(\hat{x}_{(w)}^n) = z(\hat{x}_{(w)}^{t_1}) + \sum_{t_1 \le t \le n} z(\hat{x}_{(w)}^t) - z(\hat{x}_{(w)}^{t-1}).$$

Taking expectations, by Lemma 2.5 (which is still valid here) we have:

$$\mathbf{E}\left[z(\hat{x}_{(w)}^t) - z(\hat{x}_{(w)}^{t-1})\right] \le$$

$$2^{r+2}n^r/t^r + 2\frac{n}{t(n-t+1)}\mathbf{E}\left[|b(\hat{x}_{(w)}^{t-1}) - b(\frac{n}{t}x_{(w)}^{t-1})|_1\right].$$

Now, note that we have

$$|b(\hat{x}_{(w)}^{t-1}) - b(\frac{n}{t}x_{(w)}^{t-1})|_1 \le \max_{y \in Y} |b(\hat{x}_{(y)}^{t-1}) - b(\frac{n}{t}x_{(y)}^{t-1})|_1.$$

Thus we can use Lemma 2.22 to deal with this term. To analyze $z(\hat{x}_{(w)}^{t_1})$, we write:

$$z\left(\hat{x}_{(w)}^{t_1}\right) \le z\left(\frac{n}{t_1}x_{(w)}^{t_1}\right) + \left|z\left(\hat{x}_{(w)}^{t_1}\right) - z\left(\frac{n}{t_1}x_{(w)}^{t_1}\right)\right|.$$

The second term can be bounded above by $\max_{y \in Y} |z(\hat{x}_{(y)}^{t_1}) - z(\frac{n}{t_1}x_{(y)}^{t_1})|$, so that we can apply Lemma 2.23 to bound it. As to the first term, since x^* is one of the cuts in Y, by definition of the algorithm we have

$$z\left(\frac{n}{t_1}x_{(w)}^{t_1}\right) \le z\left(\frac{n}{t_1}x_{(x^*)}^{t_1}\right).$$

Now we can write

$$z\left(\frac{n}{t_1}x^{t_1}_{(x^*)}\right) \leq z\left(\hat{x}^{t_1}_{(x^*)}\right) + \left|z\left(\hat{x}^{t_1}_{(x^*)}\right) - z\left(\frac{n}{t_1}x^{t_1}_{(x^*)}\right)\right|.$$

Again, the second term can be bounded above by $\max_{y\in Y}|z(\hat{x}^{t_1}_{(y)}) - z(\frac{n}{t_1}x^{t_1}_{(y)})|$, and we can apply Lemma 2.23 to bound it. Now, by Lemma 2.7 for $t = t_1$, we have

$$\mathbf{E}\left[z(\hat{x}^{t_1}_{(x^*)})\right] - \mathbf{E}\left[z(\hat{x}^{t_0}_{(x^*)})\right] = O(\epsilon n^2).$$

Finally, as in the previous section, it holds that $\mathbf{E}\left[z(\hat{x}^{t_0}_{(x^*)})\right] = \text{OPT}$. Together, these bounds prove Theorem 2.2.

2.5 Proof of Theorem ??

With these techniques in hand, our proof of Theorem **??** Alon et al. [2002] is actually quite simple. Recall Alon et al. [2002] that the hard direction is showing that the subproblem isn't easier than the overall problem: $\mathbf{E}\left[OPT_{S^{t_1}}\right]/t_1^2 \geq OPT/n^2 - O(\epsilon)$. The easy direction is a consequence of Lemmas 2.7 and 2.23 (though much easier proofs exist). The key idea we use is that while Algorithm **??**, as a subroutine of Algorithm 2.2, is solving the problem on the first t_1 vertices, it is also implicitly generating solutions for the whole graph $\hat{x}^{t_1}_{(y)}$.

By Theorem **??** the best cut found by Algorithm **??** for the outer sample S^{t_1} is a good approximation to the optimal cut for the outer sample:

$$\mathbf{E}\left[z(x^{t_1}_{(w)})\right]/t_1^2 \leq \mathbf{E}\left[OPT_{S^{t_1}}\right]/t_1^2 + O(\epsilon)$$

By Lemma 2.22, the cost of the best cut of the outer sample found by Algorithm **??** is approximately equal to the cost of the extrapolated solution $\hat{x}^{t_1}$:

$$\mathbf{E}\left[z(\hat{x}^{t_1}_{(w)})\right]/n^2 \leq \mathbf{E}\left[z(x^{t_1}_{(w)})\right]/t_1^2 + O(\epsilon)$$

Recall that $\hat{x}$ has every variable set to a convex combination of several values ($|\hat{x}_u| = 1$). One can make greedy changes to convert $\hat{x}$ into an integral solution, so by definition of OPT:

$$OPT/n^2 \leq \mathbf{E}\left[z(\hat{x}^{t_1}_{(w)})\right]/n^2$$

Adding inequalities together yields:

$$OPT/n^2 \leq \mathbf{E}\left[OPT_{S^{t_1}}\right]/t_1^2 + O(\epsilon).$$

2.6 Proof of Theorem ??

2.7 Open problems

It is well-known that running greedy once is a 2-approximation for MaxCut on general graphs, and we just proved that running greedy many times is a $(1+\epsilon)$- approximation on dense graphs. Does repeating greedy on general graphs yield an approximation factor better than 2?

Previous algorithms for MaxCut have been extended to weighted instances when the weights define a metric. We conjecture that such instances can also be solved by an extension of our greedy algorithms.

What problems other than Max-r-CSP can be analyzed with our technique? It is easy to see that maximum separator can be solved with our techniques (albeit with a less efficient $t_0 = 1/\epsilon^4$), but many other possibilities exist for which PTASs are not already known. For example, consider the problem of finding a maximum cut with the condition that the fraction of the *edges* that are within the left side of the cut is the inverse of a prime (1/2, 1/3, 1/5, etc.). This problem is likely also solvable by our framework. Non-convex but smooth objectives should also be handled by our framework, such as finding a three-way cut that maximizes the square of the number of edges between piece 1 and 2 plus the number of edges between 2 and 3, should anyone care for such an objective function.

Chapter 3

Fragile MIN-k-CSPs

3.1 Introduction

The results presented in this chapter are joint work with Marek Karpinski. They previously appreared in Karpinski and Schudy [2009a].

MIN-kCSPs can be much harder to approximate than MAX-kCSPs since the error must be charged against a potentially much smaller optimum value. For this reasons many MIN-kCSPs provably have no PTAS even for everywhere-dense instances. In fact even determining whether the optimum if zero or not can be NP-hard. One such problem is the everywhere-dense version of the MIN-3UNSAT problem Bazgan et al. [2003].

Arora, Karger and Karpinski [Arora et al., 1995] introduced the first PTASs for dense *minimum* constraint satisfaction problems. They give PTASs with runtime $n^{O(1/\epsilon^2)}$ [Arora et al., 1995] for min bisection and multiway cut (MIN-d-CUT). Bazgan, Fernandez de la Vega and Karpinski [Bazgan et al., 2003] designed PTASs for MIN-SAT and the nearest codeword problem with runtime $n^{O(1/\epsilon^2)}$. We give linear-time approximation schemes for all of the problems mentioned in this paragraph except for the MIN-BISECTION problem.

Theorem 3.1. *For every $\epsilon > 0$ there is a randomized $1+\epsilon$-approximation algorithm for everywhere-dense fragile MIN-kCSPs with runtime $O(n^k) + 2^{O(1/\epsilon^2)}$.*

3.2 Intuition

We use the Gale-Berlekamp Switching Game (GB Game) as a concrete context to describe our techniques. The GB Game was introduced independently by at least three groups. The eponymous discoveries were by Elwyn Berlekamp [Carlson and Stolarski, 2004, Spencer, 1994] and David Gale [Spencer, 1994] independently introduced this problem in the context of coding theory in the 1960s. Andrew Gleason independently discovered this problem in 1958 as a sandbox for the study of

early local search methods [Gleason, 1960].[1] The GB game is played using of a m by m grid of lightbulbs. The adversary chooses an arbitrary subset of the lightbulbs to be initially "on." Next to every row (resp. column) of lightbulbs is a switch, which can be used to invert the state of every lightbulb in that row (resp. column). The protagonist's task is to minimize the number of lit lightbulbs (by flipping switches). This problem was recently proven to be NP-hard [Roth and Viswanathan, 2008]. Let $\Phi = \{-1, 1\} \subset \mathbb{R}$. For matrices M, N let $d(M, N)$ denote the number of entries where M and N differ. It is fairly easy to see that the GB Game is equivalent to the following natural problems: [Roth and Viswanathan, 2008]

- Given matrix $M \in \Phi^{m \times m}$ find column vectors $x, y \in \Phi^m$ minimizing $d(M, xy^T)$.
- Given matrix $M \in \mathbb{F}_2^{m \times m}$ find $x, y \in \mathbb{F}_2^m$ minimizing $\sum_{ij} \mathbb{1}\left(M_{ij} \neq x_i \oplus y_j\right)$ where $\mathbb{F}_2$ is the finite field over two elements with addition operator $\oplus$.
- Given matrix $M \in \Phi^{m \times m}$ find column vectors $x, y \in \Phi^m$ *maximizing* $x^T M y$.

Consider the following scenario. Suppose that our nemesis, who knows the optimal solution to the Gale-Berlekamp problem shown in Figure 3.1, gives us a constant size random sample of it to tease us. How can we use this information to construct a good solution? One reasonable strategy is to set each variable greedily based on the random sample. Throughout this section we will focus on the row variables; the column variables are analogous. For simplicity our example has the optimal solution consisting of all of the switches in one position, which we denote by α. For row v, the greedy strategy, resulting in assignment $x^{(1)}$, is to set switch v to α iff $\hat{b}(v, \alpha) < \hat{b}(v, \beta)$, where $\hat{b}(v, \alpha)$ (resp. $\hat{b}(v, \beta)$) denotes the number of light bulbs in the intersection of row v and the sampled columns that would be lit if we set the switch to position α (resp. β).

With a constant size sample we can expect to set most of the switches correctly but a constant fraction of them will elude us. Can we do better? Yes, we simply do greedy again. The greedy prices analogous to $\hat{b}$ are shown in the columns labeled with b in the middle of Figure 3.1. For the example at hand, this strategy works wonderfully, resulting in us reconstructing the optimal solution exactly, as evidenced by the $b(x^{(1)}, v, \alpha) < b(x^{(1)}, v, \beta)$ for all v. In general this does not reconstruct the optimal solution but provably gives something close.

Some of the rows, e.g. the last one, have $b(x^{(1)}, v, \alpha)$ much less than $b(x^{(1)}, v, \beta)$ while other rows, such as the first, have $b(x^{(1)}, v, \alpha)$ and $b(x^{(1)}, v, \beta)$ closer together. We call variables with $|b(x^{(1)}, v, \alpha) - b(x^{(1)}, v, \beta)| > \Theta(n)$ *clearcut*. Intuitively, one would expect the clearcut rows to be more likely correct than the nearly tied ones. In fact, we can show that we get all of the clearcut ones correct, so the remaining problem is to choose values for the rows that are close to tied. However, those rows have a lot of lightbulbs lit, suggesting that the optimal value is large, so it is reasonable to run an additive approximation algorithm and use that to set the remaining variables.

Finally observe that we can simulate the random sample given by the nemesis by simply taking a random sample of the variables and then doing exhaustive search of all possibly assignments of those variables. We have just sketched our algorithm.

[1] I would like to thank Donald Knuth for bringing Andrew Gleason's work to my attention.

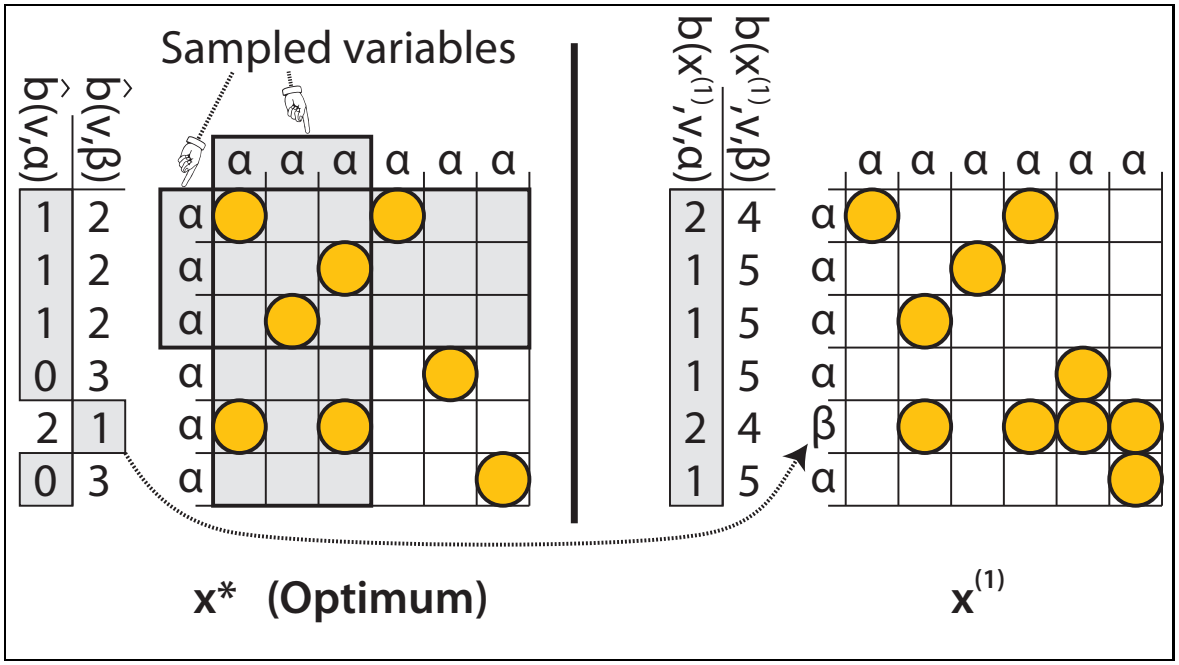

Figure 3.1: An illustration of our algorithmic ideas on the Gale-Berlekamp Game.

Our techniques differ from previous work [Bazgan et al., 2003, Arora et al., 1995, Giotis and Guruswami, 2006] in two key ways:

1. Previous work used a sample size of $O((\log n)/\epsilon^2)$, which allowed the clearcut variables to be set correctly after a single greedy step. We instead use a constant-sized sample and run a second greedy step before identifying the clearcut variables.

2. Our algorithm is the first one that runs the additive error algorithm after identifying clearcut variables. Previous work ran the additive error algorithm at the beginning.

The same ideas apply to all dense fragile CSPs.

3.3 Model

We now give a formulation of MIN-kCSP that is suitable for our purposes. (Our notation in this chapter is of (2.7) in Section **??**.) For non-negative integers n, k, let $\binom{n}{k} = \frac{n!}{k!(n-k)!}$, and for a given set V let $\binom{V}{k}$ denote the set of subsets of V of size k (analogous to 2^S for all subsets of S). There is a set V of n variables, each of which can take any value in constant-sized domain D. Let $x_v \in D$ denote the value of variable v in assignment x.

Consider some set of variables $I \in \binom{V}{k}$. There may be many constraints over these variables; number them arbitrarily. Define $p(I, \ell, x)$ to be 1 if the ℓth constraint over I is unsatisfied in assignment x and zero otherwise. For $I \in \binom{V}{k}$, we define $p_I(x) = \frac{1}{\eta} \sum_\ell p(I, \ell, x)$, where η is a scaling factor to ensure $0 \leq p_I(x) \leq 1$ (e.g. $\eta = 2^k$ for MIN-kSAT). For notational simplicity we write p_I as

a function of a complete assignment, but $p_I(x)$ only depends on x_u for variables $u \in I$. For $I \notin \binom{V}{k}$ define $p_I(x) = 0$.

Definition 3.2. *On input V, p a* minimum constraint satisfaction problem *(MIN-kCSP) is a problem of finding an assignment x minimizing $Obj(x) = \sum_{I \in \binom{V}{k}} p_I(x)$.*

Let $R_{vi}(x)$ be an assignment over the variables V that agrees with x for all $u \in V$ except for v where it is i; i.e. $R_{vi}(x)_u = \begin{cases} i & \text{if } u = v \\ x_u & \text{otherwise} \end{cases}$. We will frequently use the identity $R_{vx_v}(x) = x$. Let $b(x, v, i) = \sum_{I \in \binom{V}{k} : v \in I} p_I(R_{vi}(x))$ be the number of unsatisfied constraints v would be in if x_v were set to i (divided by η).

We say the ℓth constraint over I is *fragile* if $p(I, \ell, R_{vi}(x)) + p(I, \ell, (R_{vj}(x)) \geq 1$ for all assignments x, variables v and distinct values i and j.

Definition 3.3. *A Min-kCSP is* δ-fragile-dense *(or simply* fragile-dense*) for some $\delta > 0$ if $b(x, v, i) + b(x, v, j) \geq \delta \binom{n}{k-1}$ for all assignments x, variables v and distinct values i and j.*

Lemma 3.4. *For any $\delta > 0$ an instance where every variable $v \in V$ participates in at least $\delta\eta \binom{n}{k-1}$ fragile constraints is δ-fragile-dense.*

Proof. By definitions:

$$\begin{aligned} & b(x, v, i) + b(x, v, j) \\ = & \sum_{I \in \binom{V}{k} : v \in I} (p_I(R_{vi}(x)) + p_I(R_{vj}(x))) \\ = & \sum_{I \in \binom{V}{k} : v \in I} \frac{1}{\eta} \sum_{\ell} (p(I, \ell, R_{vi}(x)) + p(I, \ell, R_{vj}(x))) \\ \geq & \sum_{I \in \binom{V}{k} : v \in I} \frac{1}{\eta} \cdot (\text{The number of fragile constraints over } I) \\ \geq & \frac{\delta\eta}{\eta} \binom{n}{k-1} = \delta \binom{n}{k-1} \end{aligned}$$

□

We will make no further mention of individual constraints, η or fragility; our algorithms and analysis use p_I and the fragile-dense property exclusively.

3.4 Algorithm

Recall that $b(x^*, v, i)$ can be written as a sum, over $S \in \binom{V}{k-1}$, of $p_{S \cup \{v\}}(R_{vi}(x^*))$, which is a function of x^*_u for $u \in S$. We estimate $b(x^*, v, i)$ with a random sample of $s = \frac{18 \log(480|D|k/\delta)}{\delta^2}$ of its terms. Let $S_1, S_2, \ldots, S_s$ be independent random samples of $k-1$ variables from V. Later (see Lemma 3.10) we show that

$$\hat{b}(v, i) = \frac{\binom{n}{k-1}}{s} \sum_{j=1}^{s} p_{S_j \cup \{v\}}(R_{vi}(\hat{x^*}))$$

1: Run a $\frac{\epsilon}{1+\epsilon} \cdot \frac{\delta^2}{72k}\binom{n}{k}$ additive approximation algorithm.
2: **if** $Obj(answer) \geq \binom{n}{k}\delta^2/(72k)$ **then**
3: Return *answer*.
4: **else**
5: Let $s = \frac{18\log(480|D|k/\delta)}{\delta^2}$
6: Draw $S_1, S_2, \ldots, S_s$ randomly from $\binom{V}{k-1}$ with replacement.
7: **for** Each assignment $\hat{x^*}$ of the variables in $\bigcup_{j=1}^{s} S_j$ **do**
8: For all $v \in V$ and $i \in D$ let
$\hat{b}(v,i) = \frac{\binom{n}{k-1}}{s}\sum_{j=1}^{s} p_{S_j \cup \{v\}}(R_{vi}(\hat{x^*}))$
9: For all $v \in V$ let $x_v^{(1)} = \arg\min_i \hat{b}(v,i)$
10: For all $v \in V$ let $x_v^{(2)} = \arg\min_i b(x^{(1)}, v, i)$
11: Let $C = \{\, v \in V \;:\; b(x^{(1)}, v, x_v^{(2)}) < b(x^{(1)}, v, j) - \delta\binom{n}{k-1}/6 \text{ for all } j \neq x_v^{(2)} \,\}$.
12: Find $x^{(3)}$ of cost at most $\frac{\epsilon|V\setminus C|\delta}{3n}\binom{n}{k} + \min[Obj(x)]$ using an additive approximation algorithm, where the minimum ranges over x such that $x_v = x_v^{(2)}\ \forall v \in C$. See Section 3.6 for details.
13: **end for**
14: Return the best assignment $x^{(3)}$ found.
15: **end if**

Figure 3.2: Our $1+\epsilon$ approximation for δ-fragile-dense MIN-kCSP with variables taking values from domain D.

is an unbiased estimator of $b(x^*, v, i)$.

We now describe our linear-time Algorithm 3.2 for fragile-dense MIN-k-CSPs. We first (lines 1–3) dispose of the easy case OPT=$\Omega(\binom{n}{k})$ by running an additive error algorithm and returning the output if it is sufficiently costly. Second (lines 5–7) we take a random sample of variables. The optimal assignment x^* is of course unknown to us so we simply try every possible assignment of the sampled variables (line 7). We then compute $\hat{b}$ (line 8) and then make a preliminary assignment $x^{(1)}$ by setting each variable greedily relative to $\hat{b}$ (line 9). To reduce noise we do a second greedy step, yielding assignment $x^{(2)}$ (line 10). While constructing $x^{(2)}$ we make a note when the best value for a variable is far better than the second best; we fix such *clear-cut* variables to their values in assignment $x^{(2)}$. We finally run an additive error algorithm (line 12) to set the remaining variables.

3.5 Analysis of Algorithm 3.2

Recall from Theorem 2.2 in section 2 that there is a randomized algorithm which returns an assignment of cost Y such that $|Y - OPT| \leq \epsilon' n^k$ in runtime $O(n^k) + 2^{O(1/\epsilon'^2)}$ for any MAX-kCSP and any $\epsilon' > 0$. A trivial reduction shows that the same result applies to MIN-kCSPs as well.

Throughout the rest of the paper let x^* denote an optimal assignment.

First consider Algorithm 3.2 when the "then" branch of the "if" is taken. Choose constants appropriately so that the additive error algorithm fails with probability at most 1/10 and assume it succeeds. Let x^a denote the additive-error solution. We know $Obj(x^a) \leq Obj(x^*) + \frac{\epsilon}{1+\epsilon}P$ and $Obj(x^a) \geq P$ where $P = \binom{n}{k}\delta^2/(72k)$. Therefore $Obj(x^*) \geq P(1 - \frac{\epsilon}{1+\epsilon}) = \frac{P}{1+\epsilon}$ and hence

$Obj(x^a) \leq Obj(x^*) + \frac{\epsilon}{1+\epsilon}(1+\epsilon)Obj(x^*) = (1+\epsilon)Obj(x^*)$. Therefore if the additive approximation is returned it is a $1+\epsilon$-approximation.

The remainder of this section considers the case when Algorithm 3.2 takes the "else" branch. Define γ so that $Obj(x^*) = \gamma\binom{n}{k}$. We have $Obj(x^*) \leq Obj(x^a) < \binom{n}{k}\delta^2/(72k)$ so $\gamma \leq \delta^2/(72k)$. We analyze the $\hat{x^*}$ where we guess x^*, that is when $\hat{x^*_v} = x^*_v$ for all $v \in \bigcup_{i=1}^{s} S_i$. Clearly the overall cost is at most the cost of $x^{(3)}$ during the iteration when we guess correctly.

Lemma 3.5. *$b(x^*, v, x^*_v) \leq b(x^*, v, j)$ for all $j \in D$.*

Proof. Immediate from definition of b and optimality of x^*. □

Lemma 3.6. *For any assignment x,*

$$Obj(x) = \frac{1}{k}\sum_{v \in V} b(x, v, x_v)$$

Proof. By definitions,

$$b(x, v, x_v) = \sum_{I \in \binom{V}{k} : v \in I} p_I(R_{vx_v}(x)) = \sum_{I \in \binom{V}{k} : v \in I} p_I(x).$$

Write $Obj(x) = \sum_{I \in \binom{V}{k}} p_I(x) = \sum_{I \in \binom{V}{k}} p_I(x) \left[\sum_{v \in I} \frac{1}{k}\right]$ and reorder summations. □

Definition 3.7. *We say variable v in assignment x is* corrupted *if $x_v \neq x^*_v$.*

Definition 3.8. *Variable v is* clear *if $(x^*, v, x^*_v) < b(x^*, v, j) - \frac{\delta}{3}\binom{n}{k-1}$ for all $j \neq x^*_v$. A variable is* unclear *if it is not clear.*

Clearness is the analysis analog of the algorithmic notion of clear-cut vertices sketched in Section 3.2. Comparing the definition of clearness to Lemma 3.5 further motivates the terminology "clear."

Lemma 3.9. *The number of unclear variables t satisfies*

$$t \leq 3(n-k+1)\gamma/\delta \leq \frac{\delta n}{24k}.$$

Proof. Let v be unclear and choose $j \neq x^*_v$ minimizing $b(x^*, v, j)$. By unclearness, $b(x^*, v, x^*_v) \geq b(x^*, v, j) - (1/3)\delta\binom{n}{k-1}$. By fragile-dense, $b(x^*, v, x^*_v) + b(x^*, v, j) \geq \delta\binom{n}{k-1}$. Adding these inequalities we see

$$b(x^*, v, x^*_v) \quad \geq \quad \frac{1-1/3}{2}\delta\binom{n}{k-1} = \frac{1}{3}\delta\binom{n}{k-1} \tag{3.1}$$

By Lemma 3.6 and (3.1),

$$\begin{aligned} OPT &= \gamma\binom{n}{k} = 1/k\sum_u b(x^*, u, x^*_u) \\ &\geq 1/k \sum_{u:unclear} \frac{\delta}{3}\binom{n}{k-1} = \frac{\delta}{3k}\binom{n}{k-1}t. \end{aligned}$$

Therefore $t \leq \gamma\binom{n}{k}\frac{3k}{\delta\binom{n}{k-1}} = \frac{3\gamma}{\delta}(n-k+1)$.

For the second bound observe $3n\gamma/\delta \leq \frac{3n}{\delta}\frac{\delta^2}{72k} = \frac{\delta n}{24k}$. □

Lemma 3.10. *The probability of a fixed clear variable v being corrupted in $x^{(1)}$ is bounded above by $\frac{\delta}{240k}$.*

Proof. First we show that $\hat{b}(v,i)$ is in fact an unbiased estimator of $b(x^*,v,i)$ for all i. By definitions and in particular by the assumption that $p_I = 0$ when $|I| < k$, we have for any $1 \le j \le s$:

$$\begin{aligned}\mathbf{E}\left[p_{S_j\cup\{v\}}(R_{vi}(x^*))\right] &= \frac{1}{\binom{n}{k-1}} \sum_{J\in\binom{V}{k-1}} p_{J\cup\{v\}}(R_{vi}(x^*)) \\ &= \frac{1}{\binom{n}{k-1}} \sum_{I\in\binom{V}{k}:v\in I} p_I(R_{vi}(x^*)) \\ &= \frac{1}{\binom{n}{k-1}} b_{vi}(x^*)\end{aligned}$$

Therefore

$$\mathbf{E}\left[\hat{b}(v,i)\right] = s\frac{\binom{n}{k-1}}{s}\mathbf{E}\left[p_{S_1\cup\{v\}}(R_{vi}(x^*))\right] = b(x^*,v,i).$$

Recall that $0 \le p_I(x) \le 1$ by definition of p, so by Azuma-Hoeffding,

$$\begin{aligned}&\Pr\left(\left|\textstyle\sum_{j=1}^s p_{S_j\cup\{v\}}(R_{vi}(x^*)) - \frac{s}{\binom{n}{k-1}} b(x^*,v,i)\right| \ge \lambda s\right) \\ &\qquad \le 2e^{-2\lambda^2 s}\end{aligned}$$

for any $\lambda > 0$ hence

$$\Pr\left(|\hat{b}(v,i) - b(x^*,v,i)| \ge \lambda\binom{n}{k-1}\right) \le 2e^{-2\lambda^2 s}$$

Choose $\lambda = \delta/6$ and recall $s = \frac{18\log(480|D|k/\delta)}{\delta^2}$, yielding.

$$\Pr\left(|\hat{b}(v,i) - b(x^*,v,i)| \ge \frac{\delta}{6}\binom{n}{k-1}\right) \le \frac{\delta}{240|D|k}$$

By clearness we have $b(x^*,v,j) > b(x^*,v,x_v^*) + \delta\binom{n}{k-1}/3$ for all $j \ne x_v^*$. Therefore, the probability that $\hat{b}(v,x_v^*)$ is not the smallest $\hat{b}(v,j)$ is bounded by $|D|$ times the probability that a particular $\hat{b}(v,j)$ differs from its mean by at least $\delta\binom{n}{k-1}/6$. Therefore $\Pr\left(x_v^{(1)} \ne x_v^*\right) \le |D|\frac{\delta}{240|D|k} = \frac{\delta}{240k}$. ☐

Let E_1 denote the event that the assignment $x^{(1)}$ has at most $\delta n/12k$ corrupted variables.

Lemma 3.11. *Event E_1 occurs with probability at least $1 - 1/10$.*

Proof. We consider the corrupted clear and unclear variables separately. By Lemma 3.9, the number of unclear variables, and hence the number of corrupted unclear variables, is bounded by $\frac{\delta n}{24k}$.

The expected number of clear corrupted variables can be bounded by $\frac{\delta n}{240k}$ using Lemma 3.10, so by Markov bound the number of clear corrupted variables is less than $\frac{\delta n}{24k}$ with probability at least 1 - 1/10.

Therefore the total number of corrupted variables is bounded by $\frac{\delta n}{24k} + \frac{\delta n}{24k} = \frac{\delta n}{12k}$ with probability at least 9/10. ☐

We henceforth assume E_1 occurs. The remainder of the analysis is deterministic.

Lemma 3.12. *For assignments y and y' that differ in the assignment of at most t variables, for all variables v and values i, $|b(y,v,i) - b(y',v,i)| \leq t\binom{n}{k-2}$.*

Proof. Clearly $p_I(R_{vi}(y))$ is a function only of the variables in I excluding v, so if $I - \{v\}$ consists of variables u where $y_u = y'_u$, then $p_I(R_{vi}(y)) - p_I(R_{vi}(y')) = 0$. Therefore $b(y,v,i) - b(y',v,i)$ equals the sum, over $I \in \binom{V}{k}$ containing v and at least one variable u other than v where $y_u \neq y'_u$, of $[p_I(R_{vi}(y)) - p_I(R_{vi}(y'))]$. For any I, $|p_I(R_{vi}(y)) - p_I(R_{vi}(y'))| \leq 1$, so by the triangle inequality a bound on the number of such sets suffices to bound $|b(y,v,i) - b(y',v,i)|$. The number of such sets can trivially be bounded above by $t\binom{n}{k-2}$. □

Lemma 3.13. *Let $C = \{\, v \in V \;:\; b(x^{(1)}, v, x_v^{(2)}) < b(x^{(1)}, v, j) - \delta\binom{n}{k-1}/6$ for all $j \neq x_v^{(2)} \,\}$ as defined in Algorithm 3.2. If E_1 then:*

1. *$x_v^{(2)} = x_v^*$ for all $v \in C$.*
2. *$|V \setminus C| \leq \frac{3n\gamma}{\delta}$.*

Proof. Assume E_1 occurred. Using the definitions of corrupted and event E_1 together with Lemma 3.12 we have for any v, i (for sufficiently large n so that $\frac{n-k+1}{k-1} \geq \frac{n}{k}$):

$$\begin{aligned} |b(x^{(1)}, v, i) - b(x^*, v, i)| &\leq \frac{\delta n}{12k}\binom{n}{k-2} \\ &\leq \frac{\delta}{12}\binom{n}{k-1}. \end{aligned} \tag{3.2}$$

To prove the first statement of the Lemma, let v be a clear-cut variable, i.e. $v \in C$. Apply (3.2), yielding:

$$\begin{aligned} & b(x^*, v, x_v^{(2)}) \\ &\leq b(x^{(1)}, v, x_v^{(2)}) + \frac{\delta}{12}\binom{n}{k-1} \\ &< b(x^{(1)}, v, j) - \frac{\delta}{6}\binom{n}{k-1} + \frac{\delta}{12}\binom{n}{k-1} \\ &\leq b(x^*, v, j) + \left(-\frac{\delta}{6} + 2\frac{\delta}{12}\right)\binom{n}{k-1} = b(x^*, v, j). \end{aligned}$$

So by Lemma 3.5, $x_v^* = x_v^{(2)}$.

To prove the second statement, let u be clear and use (3.2) again:

$$\begin{aligned} b(x^{(1)}, u, x^*_u) &\leq b(x^*, u, x^*_u) + \frac{\delta}{12}\binom{n}{k-1} \\ &< b(x^*, u, j) - \frac{\delta}{3}\binom{n}{k-1} + \frac{\delta}{12}\binom{n}{k-1} \\ &\leq b(x^{(1)}, u, j) + \left(-\frac{\delta}{3} + 2\frac{\delta}{12}\right)\binom{n}{k-1} \\ &= b(x^{(1)}, u, j) - \frac{\delta}{6}\binom{n}{k-1} \end{aligned}$$

so $u \in C$ by definition of C. Therefore the second statement follows from bound on the number of unclear variables from Lemma 3.9. □

3.6 Computation of $x^{(3)}$

Now we give the details of the computation of $x^{(3)}$. Let $T = V \setminus C$. We call C the *clear-cut* vertices and T the *tricky* vertices. If $|T| \geq n/2$ the desired approximation factor can be achieved using a direct application of any additive error algorithm so suppose $|T| < n/2$.[2] It is natural to substitute the values of the clear-cut variables into the constraints, but doing these leaves constraints with arity less than k. To remedy this we create placeholder variables as follows. We divide C into $|T|$ sets $C_1, C_2, \ldots, C_{|T|}$ all of size at most $2|C|/|T|$. We create new placeholder variables $P = \{\nu_1, \ldots, \nu_{|T|}\}$ to represent the variables in $C_1, C_2, \ldots, C_{|T|}$ respectively.

We create a new CSP over variables $T \cup P$ by replacing each clear-cut variable in C with its representative in P. Constraints with insufficient arity are padded with arbitrary placeholder variables from P. Constraints that formerly depended only on variables in C can sefely be discarded.

Formally, for any $H \subseteq C$ let $\mu(H)$ denote the variables representing H, i.e. $\mu(H)$ is an arbitrary subset of P of size $|H|$ satisfying $H \subseteq \bigcup_{\nu_i \in \mu(H)} C_i$.

For any assignment y of $T \cup P$, $M \subseteq T$ and $L \subseteq P$ with $|M| + |L| = k$, define

$$q_{M,L}(y) = \mathbb{1}\left(|L| \leq k-1\right) \sum_{H \in \binom{C}{|L|} : \mu(H) = L} p_{M \cup H}(R_{Ty}(x^{(2)}))$$

The following Lemma follows easily from the definitions:

Lemma 3.14. *For any $y \in D^{2|T|}$ we have*

$$Obj(R_{Ty}(x^{(2)})) = \sum_{K \in \binom{T}{k}} q_K(y) + \sum_{I \in \binom{C}{k}} p_I(x^{(2)})$$

Lemma 3.15.

$$0 \leq q_K(y) \leq O\left(\left(\frac{|C|}{|T|}\right)^{k-1}\right)$$

[2] Alternatively Lemma 3.13 implies $|T| \leq \delta n/(24k) \leq n/2$

Proof. Let $M = K \cap T$ and $L = K \cap P$. If $|L| = k$ the Lemma is trivial so the interesting case is $|L| \leq k-1$. Recall that $\rho(H) = L$ implies $|H| = |L| \leq k-1$, so any H contributing to the sum is a subset of $\bigcup_{\nu_i \in L} C_i$, which has size at most $2|L||C|/|T|$. Therefore the number of such H is at most $\binom{2|L||C|/|T|}{k-1} = O\left(\left(\frac{|C|}{|T|}\right)^{k-1}\right)$. □

After using Lemma 3.15 to normalize the weights, Theorem 2.2 with an error parameter of $\epsilon' = \Theta(\epsilon)$ yields an additive error of $O(\epsilon|T|^k(|C|/|T|)^{k-1}) = O(\epsilon(|T|/n)n^k)$ for the problem of minimizing $\sum_{K \in \binom{T \cup P}{k}} q_K(y)$. By Lemma 3.14 this is also an additive error $O(\epsilon(|T|/n)n^k)$ for $Obj(R_{Ty}(x^{(2)}))$ as stated in the algorithm. Using Lemma 3.13 we further bound the additive error $O(\epsilon(|T|/n)n^k)$ by $O(\epsilon\gamma n^k)$. Lemma 3.13 implies that $x^* = R_{Ty}(x^{(2)})$ for some y, so this yields an additive error $O(\epsilon\gamma n^k) = \epsilon OPT$ for our original problem of minimizing $Obj(x)$ over all assignments x.

3.6.1 Runtime

In this subsection we heavily exploit our choice to hide δ, $|D|$ and k within $O(\cdot)$. Note $s = O(1)$, hence the number of iterations of the loop is $|D|^{\left|\bigcup_{j=1}^{s} S_j\right|} \leq |D|^{(k-1)s} = O(1)$, which we may safely ignore when analyzing runtime. The two calls to the additive error algorithm take, by Theorem 2.2, $O(n^k) + 2^{O(1/\epsilon^2)}$ time. Observe that $b(x, v, i)$ can be easily computed in time $O(n^{k-1})$ directly from its definition, hence $b(x^{(1)}, v, i)$ can be computed for all all v, i in $|D|nO(n^{k-1}) = O(n^k)$ time. Computing the cost of a particular assignment can also be done in $O(n^k)$ time. It is easy to see that the tasks not specifically mentioned take $O(n)$ time, so the overall runtime is $O(n^k) + 2^{O(1/\epsilon^2)}$ as claimed.

Chapter 4

Correlation Clustering with a fixed number of clusters

4.1 Introduction

The results presented in this chapter are joint work with Marek Karpinski. They previously appreared in Karpinski and Schudy [2009a].

As noted in the introduction, the input to the correlation clustering problem consists of a graph where each edge is labeled either "+" or "-". The objective is to color the vertices with d colors minimizing the number of bichromatic "+" edges plus the number of monochromatic "-" edges. This problem was introduced by Ben-Dor et al. [1999] to cluster gene expression patterns. Since then it has been applied to problems in data mining and natural language processing [McCallum and Wellner, 2004, Finkel and Manning, 2008, Elsner and Charniak, 2008]. The correlation clustering approach has several strengths. It does not require users to specify a parametric form for the clusters, nor to pick the number of clusters. Unlike fully unsupervised clustering methods, it can use training data to optimize the pairwise classifier, but unlike classification, it does not require samples from the specific clusters found in the test data. For instance, it can use messages about cars to learn a similarity function that can then be applied to messages about whales. We discuss some applications of correlation clustering in more detail when we describe our experimental results in Chapter 6.

As correlation clustering is hard to solve exactly, many previous papers focused on approximation algorithms [Bansal et al., 2004, Charikar et al., 2005, Demaine et al., 2006, Ailon et al., 2008, Swamy, 2004]. For the purpose of approximation, note that the maximization and minimization goals are not equivalent: minimizing is harder. How hard is it to minimize disagreements? A constant approximation ratio (currently 2.5) is achievable in the restricted *complete information* setting that assumes that information is available for every pair of vertices [Ailon et al., 2008, van Zuylen et al.,

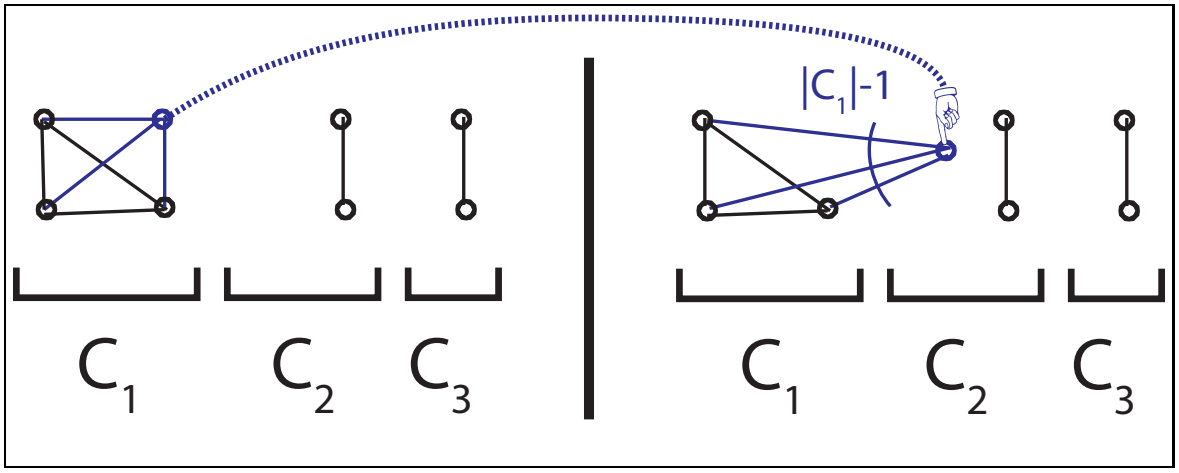

Figure 4.1: An illustration of correlation clustering and the rigidity property. Only "+" edges are shown.

2007, Bansal et al., 2004]; this result is essentially best possible (except for lowering the value of the constant) since the complete information problem is APX-hard [Charikar et al., 2005]. The generalization where no information is available for some pairs of objects admits an $O(\log n)$ approximation [Demaine et al., 2006], where n is the number of items (vertices) being clustered. In this thesis we focus on the complete information version.

How can one get around this APX-hardness result to provably get a clustering with value within $1+\epsilon$ of optimal? Additional assumptions are needed. In this section we assume that the number of clusters is a fixed constant. (In section 5.1 we study an alternate assumption, namely a noisy input model.) Correlation clustering with 2 clusters is fragile and Theorem **??** gives a linear-time approximation scheme. For $d > 2$ correlation clustering is not fragile but has properties allowing for a PTAS anyway. We also solve a generalization of correlation clustering called hierarchical clustering [Ailon and Charikar, 2005]. We prove the following theorem.

Theorem 4.1. *For every $\epsilon > 0$ there is a randomized $1+\epsilon$-approximation algorithm for correlation clustering and hierarchical clustering with fixed number of clusters d with running time $n^2 2^{O((\log d)d^6/\epsilon^2)}$.*

The above results improves on the running time $n^{O(9^d/\epsilon^2)} \cdot \log n = n^{O(9^d/\epsilon^2)}$ of the previous PTAS for correlation clustering by Giotis and Guruswami [Giotis and Guruswami, 2006] in two ways: first the polynomial is linear in the size of the input and second the exponent is polynomial in d rather than exponential. Our result for hierarchical clustering with a fixed number of clusters is the first PTAS for this problem.

4.2 Intuition

As we noted previously in Section 1, correlation clustering constraints are not fragile for $d > 2$. Indeed, the constraint corresponding to a pair of vertices that are connected by a "−" edge can be

satisfied by any coloring of the endpoints as long as the endpoints are colored differently. Fortunately there is a key observation in Giotis and Guruswami [2006] that allows for the construction of a PTAS. Consider the cost-zero clustering shown on the left of Figure 4.1. Note that moving a vertex from a small cluster to another small one increases the cost very little, but moving a vertex from a large cluster to anywhere else increases the cost a lot. Fortunately most vertices are in big clusters so, as in Giotis and Guruswami [2006], we can postpone processing the vertices in small clusters. We use the above ideas, which are due to Giotis and Guruswami [2006], the fragile-dense ideas sketched above, plus some additional ideas, to analyze our correlation clustering algorithm.

To handle hierarchical clustering (see Ailon and Charikar [2005]) we need a few more ideas. Firstly we abstract the arguments of the previous paragraph to a CSP property *rigidity*. Secondly, we note that the number of trees with d leaves is a constant and therefore we can safely try them all. We remark that all fragile-dense problems are also rigid.

4.3 Correlation and Hierarchical clustering

We now define hierarchical clustering formally (following Ailon and Charikar [2005]). For integer $M \geq 1$, an *M-level hierarchical clustering* of n objects V is a rooted tree with the elements of V as the leaves and every leaf at depth (distance to root) exactly $M + 1$. The special case $M = 1$ will turn out to be correlation clustering described in the introduction. We call the subtree induced by the internal nodes of a M-level hierarchical clustering the *trunk*. We call the leaves of the trunk *clusters*. A hierarchical clustering is completely specified by its trunk and the parent cluster of each leaf.

For a fixed hierarchical clustering and clusters i and j, let $f(i, j)$ be the distance from the root to the least common ancestor of i and j. For example when $M = 1$, $f(i, j) = \mathbb{1}(i \neq j)$.

Definition 4.2. *The M-hierarchical clustering problem with d clusters has input a set of vertices V and a similarity function F from pairs of vertices to $[0, 1]$.*[1] *The objective is to find a M-level hierarchical clustering with at most d clusters minimizing*

$$\sum_{u,v} \frac{1}{M} |F(u, v) - f(parent(u), parent(v))|.$$

We now illustrate the connection between hierarchical clustering and the correlation clustering we previously defined (informally). We now argue that the hierarchcical clustering problem with $M = 1$ is equivalent to correlation clustering. A 1-hierarchical clustering has one node at the root, some "cluster" nodes in the middle level and all of V in the bottom level. Identifying the nodes in the middle level with clusters of V we see a one-to-one correspondence between 1-hierarchical clusterings and ordinary clusterings (partitions). The hierarchical clustering objective function reduces to the number of vertex pairs u, v that are clustered together but have $F(u, v) = 1$ plus the number of vertex pairs u, v that are clustered apart but have $F(u, v) = 0$. If we identify $F(u, v) = 1$ with

[1] This is a slight generalization of Ailon and Charikar [2005], which restricts F to multiples of $1/M$.

the presence of a $-$ edge and $F(u,v) = 0$ with a "+" edge we see this is precisely the correlation clustering objective.

4.4 Reduction to Rigid MIN-2CSP

In this section we define the concept of a *weakly fragile* CSP and show a reduction from hierarchical clustering to a constant number of such CSPs.

Definition 4.3. *For any $\epsilon > 0$, a CSP constraint is* ϵ-weakly CSP-fragile *if .*

Lemma 4.4. *The number of possible trunks is at most* $d^{(M-1)d}$.

Proof. The trunk can be specified by giving the parent of all non-root nodes. There are at most d nodes on each of the $M-1$ non-root levels so the lemma follows. □

We now show how to reduce hierarchical clustering with a constant number of clusters to the solution of a constant number of min-2CSPs with cardinality of D equal to d. We use a variant of the notation used in Chapter 3 that is specialized for MIN-2CSPs. For vertices u, v and values i, j, let $p_{u,v}(i,j)$ be the cost of putting u in cluster i and v in cluster j. This is the same concept as p_I for the fragile case, but this notation is more convenient here. Define $b(x,v,i) = \sum_{u \in V\,:\,u \neq v} p_{u,v}(x_u, i)$, which is identical to b of the fragile-dense analysis but expressed using different notation.

Definition 4.5. *A MIN-2CSP is* rigid *if for some $\delta > 0$, all $v \in V$ and all $j \neq x^*_v$*

$$b(x^*, v, x^*_v) + b(x^*, v, j) \geq \delta |\{\, u \in V \setminus \{v\} \;:\; x^*_u = x^*_v \,\}|$$

Observe that $|\{\, u \in V \setminus \{v\} \;:\; x^*_u = x^*_v \,\}| \leq |V| = \binom{|V|}{2-1}$ hence any fragile-dense CSP is also rigid.

Lemma 4.6. *If the trunk is fixed, hierarchical clustering can be expressed as a $1/M$-rigid MIN-2CSP with $|D| = d$.*

Proof. (C.f. Figure 4.1) Let D be the leaves of the trunk (clusters). It is easy to see that choosing

$$p_{u,v}(i,j) = \frac{1}{M} |f(i,j) - F(u,v)|$$

yields the correct objective function. To show rigidity, fix vertex v, define $i = x^*_v$ and $\mathcal{C}_i = \{\, u \in V \;:\; x^*_u = i \,\}$. Fix $j \neq i$ and $u \in \mathcal{C}_i \setminus \{v\}$. Clearly $|f(i,i) - f(i,j)| \geq 1$, hence by triangle inequality $|F(u,v) - f(i,i)| + |F(u,v) - f(i,j)| \geq 1$, hence $p_{u,v}(i,i) + p_{u,v}(i,j) \geq 1/M$. Summing over $u \in \mathcal{C}_i \setminus \{v\}$ we see

$$\begin{aligned} b(x^*, v, x^*_v) + b(x^*, v, j) &\geq \frac{1}{M} |\mathcal{C}_i \setminus \{v\}| \\ &= \delta\, |\{\, u \in V \setminus \{v\} \;:\; x^*_u = x^*_v \,\}| \end{aligned}$$

for $\delta = 1/M$. □

Return CC(V, blank assignment, 0)

CC(tricky variables T, assignment y of $V \setminus T$, recursion depth *depth*):

1: Find assignment of cost at most $\frac{\epsilon}{1+\epsilon} \cdot \frac{\delta^3|T|^2}{6 \cdot 72^2 |D|^3} + \min_{x: x_v = y_v \forall v \in V \setminus T} [Obj(x)]$ using an additive approximation algorithm.
2: **if** $Obj(answer) \geq \frac{\delta^3|T|^2}{6 \cdot 72^2 |D|^3}$ or $depth \geq |D| + 1$ **then**
3: Return *answer*.
4: **else**
5: Let $s = \frac{432^2 |D|^4 \log(1440|D|^3/\delta)}{2\delta^4}$
6: Draw $v_1, v_2, \ldots, v_s$ randomly from T with replacement.
7: **for** Each assignment $\hat{x^*}$ of the variables $\{v_1, v_2, \ldots, v_s\}$ **do**
8: For all $v \in T$ and i let
$\hat{b}(v,i) = \frac{|T|}{s} \sum_{j=1}^{s} p_{v_j,v}(\hat{x}^*_{v_j}, i) + \sum_{u \in V \setminus T} p_{u,v}(y_u, i)$
9: For all $v \in V$ let
$x_v^{(1)} = \begin{cases} y_v & \text{if } v \in V \setminus T \\ \arg\min_i \hat{b}(v,i) & \text{otherwise} \end{cases}$
10: For all $v \in T$ let $x_v^{(2)} = \arg\min_i b(x^{(1)}, v, i)$
11: Let $C = \{ v \in T \ : \ b(x^{(1)}, v, x_v^{(2)}) < b(x^{(1)}, v, j) - \frac{\delta|T|}{12|D|} \text{ for all } j \neq x_v^{(2)} \}$.
12: Let $T' = T \setminus C$
13: Define assignment y' by
$y'_v = \begin{cases} y_v & \text{if } v \in V \setminus T \\ x_v^{(2)} & \text{if } v \in C \\ \text{Undefined} & \text{if } v \in T \setminus C \end{cases}$.
14: If CC($T', y', depth + 1$) is the best clustering so far, update best.
15: **end for**
16: Return the best clustering found.
17: **end if**

Figure 4.2: Approximation Algorithm for Rigid MIN-2CSPs.

Lemmas 4.6 and 4.4 suggest a technique for solving hierarchical clustering: guess the trunk and then solve the rigid MIN-2CSP. We now give our algorithm for solving rigid MIN-2CSPs.

4.4.1 Algorithm for Rigid MIN-2CSP

Algorithm 4.2 solves rigid MIN-2CSPs by identifying clear-cut variables, fixing them to optimal values, and then recursing on the remaining "tricky" variables T. The recursion terminates when the remaining subproblem is sufficiently expensive for an additive approximation to suffice.

4.5 Analysis of Algorithm 4.2

4.5.1 Runtime

Theorem 4.7. *For any T, y, an assignment of cost at most $\epsilon'|T|^2 + \min_{x:x_v=y_v \forall v \in V\setminus T}[Obj(x)]$ can be found in time*
$n^2 2^{O((\log|D|)/\epsilon'^2)}$.

Proof. The problem is essentially a CSP on T variables but with an additional linear cost term for each variable. It is fairly easy to see that Algorithm 1 in Mathieu and Schudy [2008a] has error proportional to the misestimation of b and hence is unaffected by arbitrarily large linear cost terms. On the other hand, the more efficient Algorithm 2 in Mathieu and Schudy [2008a] needs to estimate the objective value from a constant-sized sample as well and hence does not seem to work for this type of problem. □

In this subsection $O(\cdot)$ hides only absolute constants and $\tilde{O}(\cdot)$ hides one factor of $\log|D|$. Algorithm 4.2 has recursion depth at most $|D|+1$ and branching factor $|D|^s$, so the number of recursive calls is at most $(|D|^s)^{|D|+1} = 2^{s(|D|+1)\log|D|} = 2^{\tilde{O}(|D|^5/\delta^4)}$. Each call spends $O(|D|n^2)$ time on miscellaneous tasks such as computing the objective value plus time required to run the additive error algorithm, which is $n^2 \cdot 2^{\tilde{O}(|D|^6/\epsilon^2\delta^6)}$ by Theorem 4.7. Therefore the runtime of Algorithm 4.2 is $n^2 2^{\tilde{O}(\frac{|D|^6}{\epsilon^2\delta^6})}$, where the $2^{\tilde{O}(|D|^5/\delta^4)}$ from the size of the recursion tree got absorbed into the $2^{\tilde{O}(\frac{|D|^6}{\epsilon^2\delta^6})}$ from Theorem 4.7. For hierarchical clustering, $\delta = 1/M$ yields a runtime of $n^2 2^{\tilde{O}(\frac{|D|^6 M^6}{\epsilon^2})} \cdot |D|^{(M-1)|D|} = n^2 2^{O(\frac{(\log|D|)|D|^6 M^6}{\epsilon^2})}$.

As noted in the introduction this improves on the runtime of $n^{O\left(\frac{9|D|}{\epsilon^2}\right)}$ of Giotis and Guruswami [2006] for correlation clustering in two ways: the degree of the polynomial is independent of ϵ and $|D|$, and the dependence on $|D|$ is singly rather than doubly exponential.

4.5.2 Approximation

We fix optimal assignment x^*. We analyze the path through the recursion tree where we always guess $\hat{x}^*$ correctly, i.e. $\hat{x}^*_v = x^*_v$ for all $v \in \{v_1, v_2, \ldots, v_s\}$. We call this the *principal path.*

We will need the following definitions.

Definition 4.8. *Variable v is m-*clear *if $b(x^*, v, x^*_v) < b(x^*, v, j) - m$ for all $j \neq x^*_v$. We say a variable is* clear *if it is m-clear for m obvious from context. A variable is* unclear *if it is not clear.*

Definition 4.9. *A variable is* obvious *if it is in cluster $\mathcal{C}$ in OPT and it is $\delta|\mathcal{C}|/3$-clear.*

Definition 4.10. *A cluster $\mathcal{C}$ of OPT is* finished *w.r.t. T if $T \cap \mathcal{C}$ contains no obvious variables.*

Lemma 4.11. *With probability at least 8/10, for any triple $(T, y, depth)$ encountered on the principle path,*

1. *$y_v = x^*_v$ for all $v \in V \setminus T$ and*

2. The number of finished clusters w.r.t. T is at least depth.

Before proving Lemma 4.11 let us see why it implies Algorithm 4.2 has the correct approximation factor.

Proof. Study the final call on the principal path, which returns the additive approximation clustering. The second part of Lemma 4.11 implies that $depth \leq |D|$, hence we must have terminated because $Obj(answer) \geq \frac{\delta^3|T|^2}{6\cdot 72^2|D|^3}$. By the first part of Lemma 4.11 the additive approximation gives error at most

$$\frac{\epsilon}{1+\epsilon}\cdot\frac{\delta^3|T|^2}{6\cdot 72^2|D|^3} + OPT.$$

so the approximation factor follows from an easy calculation. □

Now we prove Lemma 4.11 by induction. Our base case is the *root*, which vacuously satisfies the inductive hypothesis since $V \setminus T = \{\}$ and $depth = 0$. We show that if a node $(T, y, depth)$ (in the recursion tree) satisfies the invariant then its *child* $(T', y', depth+1)$ does as well. We hereafter analyze a particular $(T, y, depth)$ and assume the inductive hypothesis holds for them. There is only something to prove if a child exists, so we hereafter assume the additive error answer is *not* returned from this node and hence

$$OPT \leq Obj(answer) < \frac{\delta^3|T|^2}{6\cdot 72^2|D|^3}. \tag{4.1}$$

We now prove a number of Lemmas in this context, from which the fact that $T', y', depth+1$ satisfies the inductive hypothesis will trivially follow.

Lemma 4.12. *The number of $\delta^2|T|/(216|D|^2)$-clear variables that are corrupted in $x^{(1)}$ is at most $\delta|T|/(72|D|)$ with probability at least $1 - 1/(10|D|)$.*

Proof. Essentially the same proof as for fragile MIN-kCSP shows that $\hat{b}(v,i)$ is an unbiased estimator of $b(x^*, v, i)$. This time the Azuma-Hoeffding inequality yields

$$\Pr\left(|\hat{b}(v,i) - b(x^*,v,i)| \geq \lambda|T|\right) \quad \leq \quad 2e^{-2\lambda^2 s}$$

Choose $\lambda = \frac{\delta^2}{432|D|^2}$ and recall $s = \frac{432^2|D|^4\log(1440|D|^3/\delta)}{2\delta^4}$, yielding.

$$\Pr\left(|\hat{b}(v,i) - b(x^*,v,i)| \geq \frac{\delta^2|T|}{432|D|^2}\right) \quad \leq \quad \frac{\delta}{720|D|^3}$$

By clearness we have $b(x^*, v, j) > b(x^*, v, x_v^*) + \frac{\delta^2|T|}{216|D|^2}$ for all $j \neq x_v^*$. Therefore, the probability that $\hat{b}(v, x_v)$ is not the smallest $\hat{b}(v,j)$ is bounded by $|D|$ times the probability that a particular $\hat{b}(v,j)$ differs from its mean by at least $\delta^2|T|/(432|D|^2)$. Therefore $\Pr\left(x_v^{(1)} \neq x_v^*\right) \leq |D|\frac{\delta}{720|D|^3} = \frac{\delta}{720|D|^2}$. Therefore, by Markov bound, with probability $1 - 1/(10|D|)$ the number of corrupted $\delta^2|T|/(216|D|^2)$-clear variables is at most $\delta|T|/(72|D|)$. □

There are two types of bad events: the additive error algorithm failing and our own random samples failing. We choose constants so that each of these events has probability at most $1/(10|D|)$. The principle path in the recursion tree has length at most $|D|$, so the overall probability of a bad event is at most $2/10$. We hereafter assume no bad events occur.

Lemma 4.13. *The number of $\delta c/3$-unclear variables in clusters of size at least $c+1$ is at most $\frac{6OPT}{\delta c}$.*

Proof. Let *confusing* variable refer to a $\delta c/3$-unclear variable in a cluster of size at least $c+1$. Let v be such a variable, in cluster $\mathcal{C}$ in OPT. By unclearness,

$$b(x^*, v, x_v^*) \geq b(x^*, v, j) - \delta c/3$$

for appropriate $j \neq x_v^*$ and by rigidity

$$b(x^*, v, x_v^*) + b(x^*, v, j) \geq \delta \left|\mathcal{C} \setminus \{v\}\right| \geq \delta c.$$

Adding these inequalities we see $b(x^*, v, x_v^*) \geq \delta c/3$. Therefore

$$\begin{aligned} OPT &= \frac{1}{2}\sum_v b(x^*, v, x_v^*) \geq \frac{1}{2}\sum_{v \text{ confusing}} \delta c/3 \\ &= \left|\{\, v \in T \ :\ v \text{ confusing}\,\}\right| \delta c/6 \end{aligned}$$

so $\left|\{\, v \in T \ :\ v \text{ confusing}\,\}\right| \leq \frac{6OPT}{\delta c}$. □

Lemma 4.14. *For all v, i, $|b(x^{(1)}, v, i) - b(x^*, v, i)| \leq \frac{\delta|T|}{24|D|}$*

Proof. First we show bounds on three classes of corrupted variables:

1. Lemma 4.12 bounds the number of $\frac{\delta^2|T|}{216|D|^2}$-clear corrupted variables by $\frac{\delta|T|}{72|D|}$.
2. The number of variables in clusters of size at most $\frac{\delta|T|}{72|D|^2}$ is bounded by $\frac{\delta|T|}{72|D|}$.
3. The number of $\frac{\delta^2|T|}{216|D|^2}$-unclear corrupted variables in clusters of size[2] at least $\frac{\delta|T|}{72|D|^2}+1$ is bounded by, using Lemma 4.13 and (4.1), $\frac{6OPT}{\delta}\frac{72|D|^2}{\delta|T|} \leq \frac{\delta^3|T|^2}{6\cdot 72\cdot 75|D|^3}\cdot\frac{6\cdot 72|D|^2}{\delta^2|T|} = \frac{\delta|T|}{72|D|}$.

Therefore the total number of corrupted variables in $x^{(1)}$ is at most $\frac{\delta|T|}{72|D|}+\frac{\delta|T|}{72|D|}+\frac{\delta|T|}{72|D|} = \frac{\delta|T|}{24|D|}$. The easy observation that $|b(x^{(1)}, v, i) - b(x^*, v, i)|$ is bounded by the number of corrupted variables in $x^{(1)}$ proves the Lemma. □

Lemma 4.15. *There exists an obvious variable in T that is in a cluster of size at least $|T|/2|D|$.*

Proof. We say a variable v is *confusing'* if it is non-obvious and its cluster in OPT has size at least $|T|/2|D|$. By Lemma 4.13 and (4.1)

$$\begin{aligned} \left|\{\, v \in T \ :\ v \text{ confusing'}\,\}\right| &\leq \frac{12|D|}{\delta|T|}OPT \\ &\leq \frac{12|D|}{\delta|T|}\frac{\delta^3|T|^2}{6\cdot 72^2|D|^3} < |T|/2. \end{aligned}$$

Simple counting shows there are at most $|T|/2$ variables of T in clusters of size less than $|T|/2|D|$, hence there must be an obvious variable in a big cluster. □

[2]To simplify the exposition we assume that $\frac{\delta|T|}{72|D|^2}$ is an integer. Sketch of general case: if $\frac{\delta|T|}{72|D|^2}$ is large fiddle with the constants and if $\frac{\delta|T|}{72|D|^2}$ is small T is small so one can afford to guess all possible assignments of the variables in T.

Lemma 4.16. *The number of finished clusters w.r.t. T' strictly exceeds the number of finished clusters w.r.t. T.*

Proof. Let v be the variable promised by Lemma 4.15 and $\mathcal{C}_i$ its cluster in OPT. For any obvious variable u in $\mathcal{C}_i$ note that u is $\delta|\mathcal{C}_i|/3 \geq \delta|T|/6|D|$-clear, so Lemma 4.14 implies

$$\begin{aligned} & b(x^{(1)}, u, i) \\ &\leq \quad b(x^*, u, i) + \frac{\delta|T|}{24|D|} < b(x^*, u, j) + \frac{\delta|T|}{24|D|} - \frac{\delta|T|}{6|D|} \\ &\leq \quad b(x^{(1)}, u, j) + 2\frac{\delta|T|}{24|D|} - \frac{\delta|T|}{6|D|} = b(x^{(1)}, u, j) - \frac{\delta|T|}{12|D|} \end{aligned}$$

hence u is in the set C of clear variables defined in Algorithm 4.2. Therefore, no obvious variables in $\mathcal{C}_i$ are in T' so $\mathcal{C}_i$ is finished w.r.t. T'. The existence of v implies $\mathcal{C}_i$ is not finished w.r.t. T, so $\mathcal{C}_i$ is newly finished. To complete the proof note that $T' \subseteq T$ so finished is a monotonic property. □

Lemma 4.17. *(T', y') satisfy the invariant $v \in V \setminus T' \rightarrow y'_v = x^*_v$.*

Proof. Fix $v \in V \setminus T'$. If $v \in T$ the conclusion follows from the inductive hypothesis. If $v \in T \setminus T' = C$ we need to show $y'_v = x^*_v$.

Let $i = y'_v$. For any $j \neq i$, use Lemma 4.14 to obtain

$$\begin{aligned} b(x^*, v, i) &\leq b(x^{(1)}, v, i) + \frac{\delta|T|}{24|D|} \\ &< b(x^{(1)}, v, j) + \frac{\delta|T|}{24|D|} - \frac{\delta|T|}{12|D|} \\ &\leq b(x^*, v, j) + 2\frac{\delta|T|}{24|D|} - \frac{\delta|T|}{12|D|} = b(x^*, v, j) \end{aligned}$$

so by optimality of x^* we have the Lemma. □

Lemmas 4.16 and 4.17 complete the inductive proof of Lemma 4.11.

Chapter 5

Correlation Clustering with Noisy Input

5.1 Introduction

The results presented in this chapter are joint work with Claire Mathieu. Most of these results previously appreared in Mathieu and Schudy [2010].

This Chapter continues the study of the correlation clustering problem begun in Chapter 4. As noted in Chapter 4 correlation clustering has no PTAS unless P=NP. In Chapter 4 we worked around this hardness result by assuming that the number of clustes is fixed. In this section, motivated by the data mining context, we take a different approach and assume that the input comes from a noisy model defined as follows. To generate the input, start from an arbitrary partition $\mathcal{B}$ of the vertices into clusters (*base clustering*). Then, each pair of vertices is perturbed independently with probability p. In the *fully-random* model, the input is generated from $\mathcal{B}$ simply by switching every perturbed pair. In the *semi-random* variant, an adversary controls the perturbed pairs and decides whether or not to switch them. Such noisy models are hardly new: they have been studied for many graph problems such as complete information feedback arc set [Braverman and Mossel, 2008], maximum bisection [Boppana, 1987], k-coloring, unique games[Kolla and Tulsiani, 2008], maximum clique [Jerrum, 1992, Kucera, 1995, Alon et al., 1998, Feige and Krauthgamer, 2000], and even for correlation clustering itself [Ben-Dor et al., 1999, Bansal et al., 2004, McSherry, 2001, Shamir and Tsur, 2007, Elsner and Schudy, 2009]. Indeed, studying the noisy model theoretically led [Ben-Dor et al., 1999] to a heuristic algorithm that they used to successfully cluster *real* gene expression data.

We note that, prior to our work, noisy correlation clustering had only been studied in the fully-random model [Ben-Dor et al., 1999, Bansal et al., 2004, McSherry, 2001, Shamir and Tsur, 2007]. Typically, results assume that all clusters have size bounded below, by $\Omega(n)$ [Ben-Dor et al., 1999] or by $\Omega(\sqrt{n \log n}/(1/2-p)^{1+\epsilon})$ [Shamir and Tsur, 2007]. From a slightly different angle, [Bansal et al.,

2004] makes no assumption on minimum input cluster size but finds all clusters of the base clustering that have size $\Omega(p\sqrt{n\log n}/(1/2-p)^2)$.[1,2] Their algorithm can be used to yield a $1+o(1)$ approximation for $p=\omega((\log n)/n)$. (More precisely, the additive error is $O(pn^{3/2}\sqrt{\log n}/(1/2-p)^2)$ for $\sqrt{\log n/n}<p<1/2$ and $O(n\log n)$ for smaller p.)

In contrast to our work, none of those prior results on noisy correlation clustering *certify* optimality of the output clustering. Certification is a highly desirable feature, as explained by Feige and Krauthgamer [Feige and Krauthgamer, 2000]: *"since average case algorithms do not have an a priori guarantee on their performance, it is important to certify that the algorithm is indeed successful on the particular instance at hand."* Our analysis relies on the validity of the noisy model,[3] that is also important for us. All of our results address that issue and come with accompanying certificates.

Our results. We design a simple approximation algorithm. It uses a semi-definite programming relaxation and, for rounding, a constant factor approximation algorithm [van Zuylen et al., 2007, Ailon et al., 2008]. Note that this semidefinite program was already used by [Swamy, 2004], but our rounding algorithm is completely different from theirs. We then proceed to prove that it has several desirable features.

Our first result bounds the cost of the output (Theorem 5.1). In the worst-case model, that cost is a constant-factor approximation. In the noisy model,[4] let OPT (resp. OPT_{full}) denote the optimum cost in the semi-random model (resp. fully-random model). In the fully random model, our algorithm is a PTAS. In the semi-random variant, our algorithm yields cost at most $OPT+O(n^{-1/6})OPT_{full}$ with high probability (over the random perturbation). Our algorithm also produces a lower-bound, proving that the random perturbation was "random enough" for this high probability event to occur.

Our second result (Theorem 5.2) shows one circumstance in which the base clustering can be reconstructed *exactly*. We show that if $p\leq 1/3$ and all clusters have size $\Omega(\sqrt{n})$ then the base clustering is the unique optimum solution of the natural SDP relaxation of correlation clustering, hence is the output of the algorithm.

Our third result (Theorem 5.3) analyzes the small noise regime, $p=O(n^{-\delta})$ for some δ. (Such a regime makes sense in applications in which perturbations are rare, for example if they correspond to mutations). Then, we have an algorithm which finds all clusters of size $\Omega(1/\delta)$, even in the semi-random model. It also produces a certificate (witness) proving that the clusters found are part of all optimal clusterings. For example, consider a base clustering consisting of clusters of (large enough) constant size, and let $p=n^{-1/10}$. In the input graph, every vertex belongs to, on average, $\Theta(1)$ edges inherited from the base clustering, and $\Theta(n^{9/10})$ edges created by the noisy process. Yet we can reconstruct the base clustering *exactly*, hence, for every vertex, determine which $\Theta(1)$ edges are

[1] A straightforward variation works in the semi-random model but only finds clusters of size $\Omega(np^2\sqrt{\log n}/(1/2-p)^2)$.

[2] McSherry [McSherry, 2001] also solved this problem, however their results depend on the number of clusters and are worse than [Shamir and Tsur, 2007] if the number of clusters is large.

[3] The algorithm is well-defined in general, but its quality relies on the noisy model assumption.

[4] Assuming $p\leq 1/2-n^{-1/3}$

correct among the sea of $\Theta(n^{9/10})$ noisy edges.

Our analysis uses probabilistic and combinatorial arguments, spectral properties of random matrices, and semidefinite programming duality.

Comparison between our results and previous results.

1. We give the first algorithm that achieves a $1 + o(1)$ approximation even when there are arbitrarily few noisy edges (so that OPT is small) and clusters are arbitrarily small (so that reconstructing them exactly is impossible.) (Theorem 5.1)

2. In the semi-random model, as it turns out, for constant p our additive error is $O(n^{3/2})$; the previous best was $O(\epsilon n^2)$ [Giotis and Guruswami, 2006].

3. In the fully random model, compared to previous work [Bansal et al., 2004, Shamir and Tsur, 2007] we find clusters a factor $\sqrt{\log n}$ smaller for constant p. (For example, when $p = 1/3$ we reconstruct the base clustering when all clusters are size $O(\sqrt{n})$ rather than $O(\sqrt{n \log n})$.) We improve additive error by the same factor.

4. Not only does the output have cost that, with high probability, is within $1 + o(1)$ of optimal, but our algorithm also produces a deterministic lower-bound witnessing that fact. In other words our algorithm knows whether or not its input is "sufficiently random."

5. Let $\delta > 0$. If $p = O(n^{-\delta})$ we give the first algorithm that exactly reconstructs every cluster of size $\Omega(1/\delta)$.

Our work is worse than previous work in two ways: First, the algorithm for item 5 has impractical runtime even for modest $\delta = 1/3$. Second, when p is a constant, we, like Shamir and Tsur [2007], can only find the base clusters exactly if *all* are size $\Omega(\sqrt{n})$. Bansal Blum Chawla [Bansal et al., 2004]'s result is incomparable, since they find the clusters of size $\Omega(\sqrt{n \log n})$, even if the base clustering is a mixture of large and small clusters.

5.2 Algorithms

The input is an undirected graph $G = (V, E)$, where $\{u, v\} \in E$ iff we have information that u and v are similar, and $\{u, v\} \notin E$ iff we have information that u and v are dissimilar.[5] A *clustering* is a partition of the vertices into *clusters*. To any clustering, we associate the induced graph, which has an edge between every pair of intra-cluster vertices (including self-loops), and an associated matrix, the adjacency matrix of the induced graph. We use the terminology *vertex pair* (VP) to avoid ambiguity with *edge*, which refers to vertex pairs that have an edge between them in some graph.

[5] Our model requires complete information so we use absence of an edge to represent dissimilarity, not an edge labeled "-" as in previous work.

Input: graph $G = (V, E)$
Output: clustering *Output*

Let $\widehat{E}$ be E with edges that are not part of triangles removed.
Call algorithm SDPCluster on input $(V, \widehat{E})$, yielding clustering $\mathcal{A}$.
Let U denote the vertices in singleton clusters of $\mathcal{A}$.
Use maximum matching to compute an optimal clustering of U into clusters of size at most 2.

Figure 5.1: MainCluster algorithm

The *correlation clustering problem* consists of finding a clustering $\mathcal{A}$ of V minimizing $d(E, \mathcal{A})$, where $d(\cdot, \cdot)$ denotes the Hamming distance between two graphs (viewed as sets of edges), or equivalently, half of the ℓ_1 distance between the associated adjacency matrices: for symmetric matrices M and N, $d(M, N) = \frac{1}{2}\sum_{u,v} |M_{uv} - N_{uv}|$.

Fix the error parameter p. Our noisy model assumes that the input graph $G = (V, E)$ is generated by perturbing some unknown base clustering $\mathcal{B}$ as follows: identify $\mathcal{B}$ with the associated graph. Then, in the *fully random model*, for each pair of vertices, the information is flipped independently with probability p (in other words, to go from $\mathcal{B}$ to the input graph we flip every edge and every non-edge independently with probability p). In the *semi-random model*, for each pair of vertices, the information is corrupted (noisy) independently with probability p; then, an adversary generates the input graph by choosing similarity/dissimilarity information arbitrarily for each corrupted pair of vertices (when the adversary always flips the information of every corrupted pair, the resulting input is exactly the input generated in the fully random model).

We let $\mathcal{B}$ denote the unknown base clustering, $G = (V, E)$ the input graph generated from $\mathcal{B}$ in the semi-random model, and $G_{full} = (V, E_{full})$ the corresponding graph generated from $\mathcal{B}$ in the fully-random model. OPT denotes the cost of the optimal solution for input G, and, in the semi-random model, OPT_{full} denotes the cost of the optimal solution for the associated fully-random input.

Throughout the paper, when we write that an event occurs *with high probability* (w.h.p.), we mean that it holds with probability at least $1 - n^{-\alpha}$ for some $\alpha > 0$ (over the randomness in E_{full}). Let $\mathbf{E}_{alg}[\cdot]$ denote expectation over the random choices made by algorithm MainCluster.

Our first theorem shows that the MainCluster algorithm has three desirable properties. It is a constant-factor approximation in the adversarial model, a $1 + o(1)$ approximation in the planted model, and produces a lower bound certifying its approximation factor.

Theorem 5.1 (Main Theorem)**.** *Algorithm* MainCluster *runs in polynomial time and is such that:*

1. *For any input graph G, $\mathbf{E}_{alg}[Cost(OUT)] \leq 3.5\ OPT$.*
2. *In the semi-random model, if $p \leq 1/2 - n^{-1/3}$ then, with high probability over the noisy model, $\mathbf{E}_{alg}[Cost(OUT)] \leq OPT + O(n^{-1/6})OPT_{full}$.*

Input: graph $G = (V, F)$
Output: a clustering $\mathcal{A}$

Compute an optimal solution X^* to the following semi-definite program:

$$\min\ d(X,F)\ \text{ s.t. } \begin{cases} X_{ii} = 1 & \forall i \\ X_{ij} \geq 0 & \forall i \neq j \\ X_{ij} + X_{jk} - X_{ik} \leq 1 & \forall i \neq j \neq k \\ X \text{ pos. semi-definite} \end{cases}$$

Let $U \leftarrow V$.
while U is non-empty **do**
  Pick a pivot vertex v uniformly at random from U
  $A \leftarrow \{v\}$
  For each vertex u of $U \setminus \{v\}$, add u to A independently with probability X^*_{uv}.
  Output the resulting cluster A, and let $U \leftarrow U \setminus A$.
end while

Figure 5.2: SDPCluster algorithm

3. *One can compute a lower bound L on OPT. In the fully random model, if $p \leq 1/2 - n^{-1/3}$ then, with high probability over the noisy model, $\mathbf{E}_{alg}[Cost(OUT)] \leq L + o(L)$.*

A 2.5-approximation algorithm was already known so the first part of Theorem 5.1 is a contribution to the understanding of the MainCluster algorithm, not to the understanding of the correlation clustering problem.

The noisy model is motivated by the view that $\mathcal{B}$ is the ground truth, so it makes sense to ask if we can, not just approximate the objective function, but actually recover $\mathcal{B}$. We give two results of this type. Our next theorem shows that SDPCluster recovers the base clustering exactly whenever all base clusters are sufficiently large.

Theorem 5.2. *In the semi-random model if $p \leq 1/3$ and all clusters have size at least $c_1\sqrt{n}$ then $\mathcal{B}$ is with high probability the unique optimum solution of the SDP used in* SDPCluster.

Our next theorem shows that when the noise is small it is possible to reconstruct all base clusters of super-constant size.

Theorem 5.3 (Large Cluster Theorem)**.** *Assume that $p \leq n^{-\delta}/60$ for some δ. Then in the semi-random model Algorithm 5.3 outputs a set of clusters $\mathcal{A}$ such that:*

1. *W.h.p. the output clusters $\mathcal{A}$ are exactly the clusters of the base clustering that have size $\geq 3150/\delta$.*

2. *The algorithm certifies that for any optimal clustering, its clusters of size $\geq 3150/\delta$ are exactly the output clusters.*

Almost all of our proofs apply identically in the fully random and semi-random models. For simplicity of exposition we only emphasize the distinction between the models where important.

Input: graph $G = (V, E)$.
Let $\mathcal{A} \leftarrow \emptyset$.
For every subset S of V of size s:
Let A be the set of vertices that have at least $|S|/2$ neighbors in S.
If all of the following conditions hold:

1. For all disjoint sets $T, U \subseteq A$, $|T| = |U| = s$, G has at least $.9|T||U|$ edges between T and U.
2. For all sets $T \subseteq A$, $U \subseteq V \setminus A$, $|T| = |U| = s$, G has at most $.1|T||U|$ edges between T and U.
3. $|A| \geq 3|S|$ and A is equal to the set of vertices that have at least $|A|/2$ neighbors in A.

Then add A to the collection $\mathcal{A}$.

If any of the following properties fails to hold then Output:"Failure".

1. The sets in $\mathcal{A}$ are disjoint.
2. For every cluster $A \in \mathcal{A}$:
   (a) Every vertex in A has at least $.9|A|$ neighbors in A.
   (b) Every vertex not in A has at most $.1|A|$ neighbors in A.
3. For every pair of disjoint vertex sets S, T with cardinalities $|S| = |T| = 6s$, such that no cluster $A \in \mathcal{A}$ intersects both S and T ($\forall A, A \cap S = \emptyset$ or $A \cap T = \emptyset$), there are at most $(.35)|S||T|$ edges between S and T.

Let $\mathcal{A}' = \{A \in \mathcal{A} : |A| \geq 45s\}$.
Output $\mathcal{A}'$.

Figure 5.3: Large Cluster Algorithm (parameterized by integer $s = \lceil 70/\delta \rceil$).

Remarks

1. A natural generalization of our planted noisy model has input generated by adding noise to an arbitrary graph G rather than a union of cliques $\mathcal{B}$. Unfortunately this "smoothed" model is hard to approximate; if the noise is less than $n^{-\delta}$ for some $\delta > 0$ this model has no PTAS unless P=NP. To see this, take an arbitrary correlation clustering instance with ℓ nodes and non-zero optimum cost, make $\ell^{1/\delta}$ copies of each, yielding a graph with $n = \ell^{1+1/\delta}$ nodes. Then add noise. The number of resulting noisy edges is $O(n^{2-\delta}) = O(\ell^{2/\delta+1-1/\delta}) = o(\ell^{2/\delta}) = O(\text{new optimum})$, so a PTAS for correlation clustering in this smoothed model would imply a PTAS for adversarial correlation clustering and hence P=NP Charikar et al. [2005]. This smoothed model may however be interesting for relatively large p such as $p = 1/\operatorname{polylog}(n)$.

2. A random graph $G(n, p)$ with $p = n^{-\delta}$ has max clique of size $\Theta(1/\delta)$ with high probability Bollobás [2001]. Therefore the size of clusters reconstructed by Algorithm 5.3 is optimal to within constant factors.

3. The *planted clique problem* is the special-case of our problem where all but one of the clusters of $\mathcal{B}$ have size one and the noise only adds edges, not removes them. The semi-random nature

of our planted model allows our model to include the planted clique problem as a special case. The best known result Feige and Krauthgamer [2000] for planted clique and constant p requires the clique to be of size $\Omega(\sqrt{n})$, which matches Theorem 5.2. Theorem 5.2 does *not* subsume previous results for planted clique Feige and Krauthgamer [2000] since it requires that *all* clusters be that large. One interesting open question is how the correlation clustering SDP behaves when some clusters are large and other small.

4. Our results on reconstructing clusters exactly and on approximating the objective function do *not* imply each other. To see that even optimal results for the reconstruction problem are insufficient for approximation, consider when $p = n^{-3/2}$ and the base clusters have size 1 and 2. In this setting is clearly impossible to reconstruct the base clustering from the input data and such a reconstruction would not be a good approximation to the objective even if it were available. To see that approximation is insufficient for reconstruction note that omitting a vertex from a base cluster costs at most the size of that cluster, which is much less than OPT in most circumstances.

5.3 Proof of Theorem 5.1

5.3.1 Analysis of SDPCluster Algorithm

The following extension of an analysis from Ailon et al. [2008] forms the starting point of our worst-case analysis of section 5.3.2 and is also used in section 5.3.5.

Theorem 5.4. *Ailon et al. [2008] Let $G = (V, F)$ be an instance of correlation clustering, and let $\mathcal{A}$ be the resulting output of algorithm* SDPCluster. *Then, for any clustering $\mathcal{C}'$, we have:*

$$\mathbf{E}_{alg}\left[d(\mathcal{A}, F)\right] \leq 2.5\ d(\mathcal{C}', F).$$

Proof. (Proof Sketch) Algorithm SDPCluster is almost identical to algorithm LP-KwikCluster from Ailon et al. [2008]. Our SDP (semi-definite program) includes all of the constraints in Ailon et al. [2008] ($X_{ij} \leq 1$ is implied by $X_{ii} = 1, X_{jj} = 1$, and positive semi-definiteness) as well as the positive semi-definite constraint. Integral clusterings satisfy the positive semi-definite constraint, so our SDP is a relaxation. The analysis from Ailon et al. [2008] applies unchanged to SDPCluster. □

The following theorem is very similar to Theorem 5.4, but shows an approximation factor relative to the fractional clustering X^* rather than relative to the input edge set F. The analysis, which uses techniques from Ailon et al. [2008], is deferred to the full version.

Theorem 5.5. *Let $G = (V, F)$ be an instance of correlation clustering, and let $\mathcal{A}$ be the resulting output of algorithm* SDPCluster. *Then, for any clustering $\mathcal{C}'$, we have:*

$$\mathbf{E}_{alg}\left[d(\mathcal{A}, X^*)\right] \leq 3\ d(\mathcal{C}', X^*).$$

Lemma 5.6. *Let u be a vertex with no neighbors in the input (V, F) to algorithm* SDPCluster. *Then u is in a singleton cluster of the clustering $\mathcal{A}$ obtained by* SDPCluster.

Proof. (Proof Sketch) Let u be a vertex with no neighbors in F. Assume that $X_{uv} > 0$ for some $v \neq u$ in some SDP feasible X. Consider the solution X' obtained from X by setting $X'_{uw} = 0$ for all w. It is easy to see that X' is also SDP feasible and has strictly better objective. So the optimum X^* must have $X^*_{uv} = 0$. Rounding therefore puts u in a singleton cluster. □

5.3.2 Proof of Theorem 5.1 (1)

Lemma 5.7. *For input $G = (V, E)$, there exists an optimal clustering $\mathcal{C}$ such that for every input edge $\{u, v\}$ that is inside a cluster C of $\mathcal{C}$ of size 3 or more, there exists a vertex $w \in C$ such that $\{u, v, w\}$ is a triangle in the input graph.*

Proof. Let $\mathcal{C}$ be an optimal clustering with the most clusters. Assume, for a contradiction, that there is a cluster $C \in \mathcal{C}$ of size 3 or more, and an input edge $\{u, v\}$ in C such that u and v have no common neighbor in C. Consider the clustering $\mathcal{C}'$ obtained from $\mathcal{C}$ by replacing C with two clusters, $\{u, v\}$ and $C \setminus \{u, v\}$. It is easy to check that $\mathcal{C}'$ is at least as good as $\mathcal{C}$, so $\mathcal{C}'$ is also optimal but has more clusters, contradicting the definition of $\mathcal{C}$. □

Lemma 5.8. *Let $\mathcal{C}$ be an optimal clustering for input $G = (V, E)$, let $\mathcal{A}$ be the output of* SDP-Cluster *on $\widehat{G} = (V, \widehat{E})$, and Output denote the clustering output by* MainCluster. *Then:*

$$d(Output, E) \leq d(\mathcal{C}, E) + d(\mathcal{A}, \widehat{E}).$$

Proof. Let $\mathcal{M}$ be the matching in the last step of algorithm MainCluster. Since $\mathcal{M}$ only merges singletons of $\mathcal{A}$ by using edges of G, we have $d(Output, E) = d(\mathcal{A}, E) - |\mathcal{M}|$. By the triangular inequality, $d(\mathcal{A}, E) \leq d(\mathcal{A}, \widehat{E}) + d(\widehat{E}, E)$.

Assume, without loss of generality, that $\mathcal{C}$ satisfies Lemma 5.7, and let $\mathcal{M}^*$ be the collection of clusters of $\mathcal{C}$ of size 2. By definition of $\widehat{G}$ and by Lemma 5.7, the edges of $G \setminus \widehat{G}$ are not in clusters on $\mathcal{C}$ of size 3 or more. So they are either in $\mathcal{M}^*$ or between different clusters of $\mathcal{C}$. Thus $d(\widehat{E}, E) \leq |\mathcal{M}^*| + d_1(\mathcal{C}, E)$, where $d_1(\cdot)$ only takes into account edges that are in $G \setminus \widehat{G}$.

Let U denote the vertices that are in singleton connected components of $\widehat{G}$ and hence by Lemma 5.6 in $\mathcal{A}$. Let $\mathcal{M}^*_1$ be the clusters of $\mathcal{M}^*$ contained in U, and $\mathcal{M}^*_2 = \mathcal{M}^* \setminus \mathcal{M}^*_1$. By maximality of $\mathcal{M}$, we have $|\mathcal{M}^*_1| \leq |\mathcal{M}|$.

Let $\{u_i, v_i\}$, $1 \leq i \leq |\mathcal{M}^*_2|$ denote the edges of $\mathcal{M}^*_2$, with $v_i \notin U$. Since v_i is in some non-singleton connected component $\widehat{G}$, there is an edge $\{v_i, w_i\}$ of $\widehat{G}$. By definition of $\widehat{G}$, there is a vertex z_i such that $\{v_i, w_i, z_i\}$ is a triangle of $\widehat{G}$. We *mark* the two edges $\{v_i, w_i\}$ and $\{v_i, z_i\}$. When we have done this for every i, it is easy to see that each edge of $\widehat{G}$ has at most two marks, so $|\mathcal{M}^*_2|$ is less than or equal to the total number of marked edges.

By construction, the marked edges go between different clusters of $\mathcal{C}$, but they are edges of G which are also in $\widehat{G}$, so the number of marked edges is at most $d_2(\mathcal{C}, E)$, where $d_2(\cdot)$ only counts edges that are in $\widehat{G}$.

Putting all inequalities together and noticing that $d_1(\mathcal{C}, E)+d_2(\mathcal{C}, E) = d(\mathcal{C}, E)$ yields the lemma. □

Proof. (Of Part 1 of Theorem 5.1) By Lemma 5.8 we have $d(Output, E) \leq OPT + d(\mathcal{A}, \widehat{E})$. Consider the clustering $\mathcal{C}'$ obtained from $\mathcal{C}$ by splitting clusters of size 2 whenever the corresponding edge is not in $\widehat{E}$. By Theorem 5.4 applied to $G = (V, \widehat{E})$ we have $d(\mathcal{A}, \widehat{E}) \leq 2.5d(\mathcal{C}', \widehat{E})$. Lemma 5.7 implies that $d(\mathcal{C}', \widehat{E}) \leq d(\mathcal{C}, E) = OPT$. Putting these inequalities together yields the theorem. □

5.3.3 Lower Bound

In this section we prove the following simple lower bound:

Theorem 5.9. *There exists c_2 such that if $c_2/n \leq p \leq 1/3$ then w.h.p. $OPT_{full} = \Omega(n^2p)$. If $p \leq c_2/n$ then w.h.p. either $OPT_{full} = 0$ or $OPT_{full} = \widetilde{\Omega}(n^3p^2 + nn_2p)$, where n_2 is the number of vertices in base clusters of size 2 or more.*

Lemma 5.10 (Variant of Joachims and Hopcroft [2005])**.** *$d(OPT, \mathcal{B}) \leq \frac{4n \log n}{-\log 4p}$ with high probability*

Proof. Fix some clustering $\mathcal{A}$. Let $D = d(\mathcal{B}, \mathcal{A})$. Clearly $\mathcal{A}$ is better than $\mathcal{B}$ if and only if at least $D/2$ of the D pairs where they differ are noisy. This occurs with probability at most $\binom{D}{D/2}p^{D/2} \leq 2^D p^{D/2} = (4p)^{D/2}$.

Take a union bound over all $O(n^n)$ clusterings Joachims and Hopcroft [2005] $\mathcal{A}$ with $D = d(\mathcal{B}, \mathcal{A}) \geq \frac{4n \log n}{-\log 4p}$, we bound the probability that $d(OPT, \mathcal{B}) \geq 4n\frac{\log n}{-\log 4p}$ by $n^n \cdot (4p)^{D/2} = \exp(n \log n + \frac{2n \log n}{-\log 4p} \log(4p)) = n^{-n}$. □

Proof. (Of Theorem 5.9) First we prove that $OPT = \Omega(n^2p)$ when $c_2/n \leq p \leq 1/5$. In the upper portion of this range where $1/\sqrt{n} \leq p \leq 1/5$ we see $d(OPT, \mathcal{B}) = O(n \log n)$ by Lemma 5.10. The triangle inequality and a trivial Chernoff bound yield $OPT = d(\mathcal{C}, E) \geq d(\mathcal{B}, E) - d(\mathcal{C}, \mathcal{B}) = \Omega(n^2p) - O(n \log n) = \Omega(n^2p)$. In the lower portion where $c_2/n \leq p \leq 1/\sqrt{n}$ Lemma 5.10 implies $d(OPT, \mathcal{B}) = O(n)$. As before we get $OPT = d(\mathcal{C}, E) \geq d(\mathcal{B}, E) - d(\mathcal{C}, \mathcal{B}) = \Omega(n^2p) - O(n) = \Omega(n^2p)$ for sufficiently large c_2 and hence p.

Second observe that if $n^3p^2 + nn_2p = \tilde{O}(1)$ then the Theorem is trivial. We henceforth assume that $np \leq c_2$ and $n^3p^2 + nn_2p = \omega(\log n)$. We will frequently use the fact that

$$\max(n^2p, n_2) \geq \frac{n^2p + n_2}{2} = \frac{n^3p^2 + nn_2p}{2np} = \omega(\log n). \tag{5.1}$$

A classic lower-bound on OPT is the size of any collection of VP-disjoint bad triplets (triplet packing) where a bad triplet is a set $\{u, v, w\}$ such that $E_{uv} = E_{vw} = 1$ but $E_{uw} = 0$. Consider the instance $\mathcal{B}'$ obtained from $\mathcal{B}$, by flipping every vertex pair with probability $p/2$, analogously to how E is formed by flipping vertex pairs with probability p. Note that $\widehat{E}$ can be expressed as applying noise of approximately $p/2$ to $\mathcal{B}'$. We will use $\mathcal{B}'$ to construct a triplet packing lower bound.

We first construct a large matching M of $\mathcal{B}'$ size $\Omega(n_2 + n^2p)$ as follows.

If $n_2 \geq \min(n^2p, n/2)$ note that a max matching of $\mathcal{B}$ has size at least $n_2/3$, which is at least (the smaller of $n/6$ and) $\omega(\log n)$ using (5.1). With high probability only a $O(p)$ fraction of these edges are not in $\mathcal{B}'$ so M has size $\Omega(n_2) = \Omega(n_2 + n^2p)$ w.h.p.

In the case that $n_2 < \min(n^2p, n/2)$ we find a matching among the at least $n/2$ vertices that are singletons in $\mathcal{B}$. We have $n^2p = \omega(\log n)$ by (5.1), so by a Chernoff bound the number of noisy edges is $\Omega(n^2p)$. Consider generating these noisy edges one by one, adding each to the matching M if possible. As long as $|M| \leq n/8 = \Omega(n^2p)$ (otherwise we are done) each new edge has probability at least $(1/2)^2$ of being added to the matching. These events are not independent but with a little technical effort one can create related events that are. Another Chernoff bound yields $|M| = \Omega(n^2p) = \Omega(n_2 + n^2p)$ w.h.p.

Label the vertices $V = \{v_1, v_2 \ldots v_n\}$ so that (v_{2i-1}, v_{2i}) is in the matching M for all $1 \leq i \leq |M|$. Let $\alpha = \min(|M|, n/4)$. Note that $\alpha = \Omega(n_2 + n^2p)$. We construct a triplet packing as follows. For each iteration $1 \leq i \leq \alpha$ check if there exists some $n/2 < j \leq n$ such that (v_{2i-1}, v_{2i}, v_j) is a bad triplet. If so, pick an arbitrary such j and add triplet (v_{2i-1}, v_{2i}, v_j) to the triplet packing.

Consider the probability that iteration i produces a triplet for some $1 \leq i \leq \alpha$. By construction v_{2i-1} and v_{2i} are in the same cluster of $\mathcal{B}'$. Each vertex v_j with $n/2 < j \leq n$, whether in the same base cluster or different, has probability $\Theta(p)$ of forming a bad triplet. Therefore the probability that iteration i produces a bad triplet is $\Theta(\min(np, 1)) = \Theta(np)$. The expected number of triplets packed is therefore $\Theta(np)|\alpha| = \Theta(nn_2p + n^3p^2) = \omega(\log n)$. The iterations are independent so a Chernoff bound implies $OPT = \Omega(n^3p^2 + nn_2p)$ with high probability as well. □

5.3.4 Probabilistic Lemmas

The following Lemma is a trivial application of Chernoff bounds and will be used frequently.

Lemma 5.11. *Let Z be a sum of independent indicator random variables. We have $Z \leq 2\mathbf{E}[Z] + 9\ln n$ w.h.p. If $\mathbf{E}[Z] \geq 20\log n$ then $\frac{1}{2}\mathbf{E}[Z] \leq Z \leq 2\mathbf{E}[Z]$ with high probability.*

Lemma 5.12. *W.h.p. $E_{uv} = \widehat{E}_{uv}$ for all u, v within any base cluster of size at least $6\lg n$, where E and $\widehat{E}$ are as defined in Algorithm* MainCluster.

In other words, edges within base clusters of size at least $6\lg n$ are unaffected by the first line of MainCluster.

Proof. (Of Lemma 5.12) Fix edge u, v of E,[6] with u and v both within a single base cluster of size k. For any other vertex w in the same base cluster we have $E_{uw} = E_{vw} = 1$ with probability at most $2p - p^2 \leq 3/4$. These events are independent so the edge u, v is part of no triangles with probability at most $(3/4)^{k-2}$. When $k \geq 6\lg n$ this probability is at most $2n^{-3}$, so with high probability every edge within a cluster is part of a triangle, hence in $\widehat{E}$. □

[6]Recall "edge" implies $E_{uv} = 1$

Lemma 5.13. *The expected number of edges u, v (of E) within base clusters of size at least 3 that are affected by the first line of* MainCluster *(i.e. $E_{uv} = 1$ and $\widehat{E}_{uv} = 0$) is $O(n_3 p)$, where n_3 is the number of vertices in base clusters of size 3 or more. A bound of $\tilde{O}(n_3 p + 1)$ holds with high probability.*

Proof. Each cluster of size $k \geq 3$ includes in expectation at most

$$\binom{k}{2}(2p - p^2)^{k-2} \leq \frac{k^2}{2}(2p - p^2)(3/4)^{k-3} = O(p)$$

edges that are not part of any triangles. Summing over $O(n)$ clusters proves the expectation part of the Lemma.

By Lemma 5.12 we can ignore clusters of size $\Omega(\log n)$. For cluster B_i of size between 3 and $\Theta(\log n)$ let ϵ_i denote the event that there is some edge within B_i that is not part of some triangle within B_i. These events are clearly independent and as noted above have probability $O(p)$, so by a Chernoff bound (Lemma 5.11) the number of ϵ_i that occur is $O(np + \log n)$ with high probability. Each ϵ_i contributes $O(\log^2 n)$ edges, completing the proof of the theorem. □

Lemma 5.14. *For sufficiently large constant c_3 we have with high probability all vertices are part of at most $\frac{c_3}{10}(np + \log n) - 1$ noisy pairs.*

Proof. (Proof Sketch) Semi-randomness can only help, so consider the fully random model. Fix vertex v. Each of the other vertices $u \in V$ has probability p of being a noisy pair with v, so the total number of noisy pairs is a sum of independent indicator random variables. Lemma 5.11 completes the proof. □

We say that a triplet of vertices $\{u, v, w\}$ is an *unnatural triangle* if $E_{uv} = E_{vw} = E_{uw} = 1$ but $\{u, v, w\}$ is not contained within a single base cluster.

Lemma 5.15. *The number of unnatural triangles whose vertices are all in base clusters of size at most k is $O((np)^3 + (np)^2 k)$ in expectation. A bound of $O((np)^3 + (np)^2 k + \log^6 n)$ holds with high probability.*

To prove Lemma 5.15 we use an elegant generalization of Chernoff bounds due to Kim and Vu Kim and Vu [2000], Alon and Spencer [2008] to prove the high probability portion. We present a special case of their theorem in notation suitable for our use.

Let P denote the set of all $\binom{|V|}{2}$ possible pairs of vertices $u, v \in V$. Let $\mathcal{T}$ denote a collection of triplets (sets of size 3) of vertex pairs from P. Let $\{E_p\}_{p \in P}$ denote a collection of independent indicator random variables. Let random variable Y denote the number of $T \in \mathcal{T}$ such that $E_e = 1$ for all $e \in T$. For any $A \subset P$ we have random variable Y_A equal the number of $T \in \mathcal{T}$ such that $A \subset T$ and $E_e = 1$ for all $e \in T \setminus A$.

Let $E_i = \max_{A \subset P : |A| = i} \mathbf{E}[Y_A]$. Observe that $Y = Y_{\{\}}$ and $E_0 = \mathbf{E}[Y]$.

Theorem 5.16 (Kim and Vu Kim and Vu [2000])**.** *In the scenario above if $E_1, E_2, E_3 \leq 1$ we have*

$$\Pr\left(|Y - \mathbf{E}[Y]| > a\sqrt{\max(\mathbf{E}[Y], 1)\lambda^3}\right) < d \cdot \exp(-\lambda + 2\ln n)$$

for absolute constants a and d.

Corollary 5.17. *In the scenario above if $E_1, E_2, E_3 \leq 1$ we have*

$$Y = O(\mathbf{E}[Y] + \log^6 n)$$

with high probability.

Proof. (Of Lemma 5.15) We classify unnatural triangles based on whether their vertices come from two or three distinct clusters. There are $O(n^3)$ possible triplets spanning three different clusters, and each has probability p^3 of being an unnatural triangle. Therefore the expected number of such unnatural triangles is $O(n^3p^3)$. There are $O(n^2k)$ possible triplets spanning two different clusters, and each has probability at most p^2 of being unnatural, yielding expectation $O(n^2p^2k)$.

Observe that the condition $E_3 \leq 1$ is trivially satisfied. In our applications every $T \in \mathcal{T}$ consists of the edges of a triangle, so $E_2 \leq 1$ is trivially satisfied as well.

For simplicity we assume the fully random model; the reader can readily verify that our arguments hold in the semi-random model as well. To apply Corollary 5.17 we let E be the edges in the fully random model.

Let $\mathcal{T}^{(1)}$ denote the collection of possible triangles with vertices in three distinct base clusters, where the triangles are represented by three vertex pairs from P. In order to apply Corollary 5.17 we need to bound E_1 and E_2 by 1. Recall E_1 is a maximization over sets $A \subset P$ of size 1 and fix $A = \{e\}$ for some $e \in P$. If e is within a single cluster then $Y_A = 0$. If e spans two clusters there are at most n possible $T \ni e$, each of which has probability p^2, so $E_1 \leq np^2$. Therefore by Corollary 5.17 proves the Lemma with respect to the unnatural triangles with vertices in three distinct base clusters.

Let $\mathcal{T}^{(2)}$ denote the collection of possible triangles with vertices in two distinct base clusters, where the triangles are represented by three vertex pairs from P. Fix $A = \{e\}$ for some $e \in P$. If e is within a single cluster we bound $\mathbf{E}[Y_A]$ by np^2 as before. If e spans two distinct clusters, then the third vertex to form a triangle must be one of at most $2k$ vertices. Each possible triangle has probability p, so $E_1 \leq 2kp = o(1)$. Corollary 5.16 proves the Lemma w.r.t. the unnatural triangles with vertices in two distinct base clusters. This concludes the proof of Lemma 5.15. □

5.3.5 Proof of Theorem 5.1 (2) when $p \leq n^{-2/3}$.

In this part, we analyze the algorithm in the semi-random model, assuming that $p \leq n^{-2/3}$. The case $p \geq n^{-2/3}$ is deferred to Section 5.3.6.

Lemma 5.18. *There exists c_3 such that with high probability the semi-definite program finds all base clusters B of size at least $c_3(np + \log n)$ exactly. That is, $X^*_{uv} = 1$ if $u, v \in B$, and $X^*_{uv} = 0$ if $u \in B$ and $v \notin B$.*

Proof. We choose constant c_3 equal to the constant of the same name from Lemma 5.14. We say that vertex pair u, v is *B-incident* if at least one of u and v is in B. Let X^* be an arbitrary optimal solution of the semi-definite program. Consider the solution X' obtained from X^* by modifying X^*_{uv} for B-incident vertex pairs as follows:

$$X'_{uv} = \begin{cases} X^*_{uv} & \text{if } uv \text{ not } B\text{-incident} \\ 0 & \text{if } |\{u,v\} \cap B| = 1 \\ 1 & \text{if } |\{u,v\} \cap B| = 2 \end{cases}$$

We will prove the Lemma by arguing that $X' = X^*$. Following terminology from Ailon et al. [2008], say that a triplet $\{u, v, w\}$ of vertices is a *bad triplet* if $\widehat{E}_{uv} = \widehat{E}_{vw} = 1$ and $\widehat{E}_{uw} = 0$. We enrich the semidefinite program by adding the constraint that $X_{uv} = X^*_{uv}$ for every vertex pair uv that is not B-incident. This creates a new semi-definite program (P') that has the same value as the original (so X^* is also optimal for (P')). Then, we relax the constraints by removing the constraint that X be positive semidefinite, and by only writing the ijk constraint for $\{i, j, k\}$ bad triplet such that all three vertex pairs are B-incident. Moreover, we change variables by defining $y_{uv} = \begin{cases} X_{uv} & \text{if } \widehat{E}_{uv} = 0 \\ 1 - X_{uv} & \text{if } \widehat{E}_{uv} = 1 \end{cases}$ for B-incident vertex pairs. Finally, we translate the objective function by the quantity $\sum_{uv \text{ not } B\text{-incident}} (1 - X^*_{uv})\widehat{E}_{uv} + X^*_{uv}(1 - \widehat{E}_{uv})$. We obtain the following linear program (P):

$$\begin{aligned} &\text{(P)} \\ &\min \textstyle\sum_{uv\ B\text{-incident}} y_{uv} \ \text{ s.t.} \\ &y_{uv} + y_{vw} + y_{uw} \geq 1 \ \text{ for } uvw \text{ bad triplet, } |uvw \cap B| \geq 2 \\ &y_{uv} \geq 0 \ \text{ for } uv \ B\text{-incident} \end{aligned}$$

We say that vertex pair u, v is *removed* if $\widehat{E}_{uv} \neq \mathcal{B}_{uv}$. A feasible solution to (P) is obtained by setting y'_{uv} equal to 1 if $\widehat{E}_{uv} \neq \mathcal{B}_{uv}$ and to 0 otherwise. It is clearly feasible, and its value is the number of B-incident removed pairs. We will argue that y' is the *unique* optimal solution of (P). What does that imply? Observe that y' is precisely the solution obtained from X' by our change of variables. Since it is straightforward to see that X' is feasible in (P'), and (P) is essentially a relaxation of (P'), this implies that X' is the unique optimal solution of (P'). Since X^* is optimal for (P'), we conclude that $X = X^*$, hence the Lemma.

It only remains to prove that y' is the unique optimal solution of (P). Consider the linear programming dual (D) of (P).

$$\begin{aligned} &\text{(D)} \\ &\max \textstyle\sum_T \pi_T \ \text{ s.t.} \\ &\textstyle\sum_{T \supseteq \{u,v\}} \pi_T \leq 1 \ \text{ for } uv \ B\text{-incident} \\ &\pi_T \geq 0 \ \text{ for } T \ \text{ bad triplet, } |T \cap B| \geq 2 \end{aligned}$$

A feasible solution to (D) is constructed as follows. We say that vertex pair u, v is *noisy* if $E_{uv} \neq \mathcal{B}_{uv}$. We say that a bad triplet of vertices is *broken* if at least two of the vertices from B and exactly one of the three vertex pairs are removed. We set

$$\pi_T = \begin{cases} 1/\#\{\text{broken } T' \text{ that share} \\ \text{ their removed edge with } T\} & \text{if } T \text{ is broken} \\ 0 & \text{otherwise} \end{cases}$$

Let us prove that, with high probability, π is feasible.

If vertex pair $e = uv$ is removed then, by definition of π, we have $\sum_{T:e\in T} \pi_T = 1$.

If vertex pair $e = uv$ is not removed, then any broken triplet $T = uvw$ must have either uw or vw removed, hence noisy. Using Lemma 5.14, the packing constraint associated to e therefore has

$$\begin{aligned} \sum_{T:e\in T} \pi_T &\leq \sum_{w:uw \text{ noisy}} \pi_{uvw} + \sum_{w:vw \text{ noisy}} \pi_{uvw} \\ &\leq 2\frac{c_3}{10}(np + \log n) \max_T \pi_T. \end{aligned}$$

To bound $\max_T \pi_T$, fix a broken T with removed edge $e = uv$. Observe for every vertex $w \in B\backslash\{u, v\}$, triplet $T' = uvw$ is broken, except when uw or vw is removed. By Lemma 5.12, every B-incident removed pair is also noisy, so by Lemma 5.14 we can write

$$\pi_T \leq \frac{1}{|B| - \frac{2c_3}{10}(np + \log n)} \leq \frac{10}{8c_3(np + \log n)}$$

using the assumption of Lemma 5.18 that $|B| \geq c_3(np + \log n)$. This implies $\sum_{T:e\in T} \pi_T \leq 1/4$, proving feasibility of π.

Now, observe that the value of π is exactly the number of B-incident removed pairs. This equals the value of y', so y' and π are both optimal.

Moreover, consider a non-removed pair $e = uv$. Observe that for π, the packing constraint associated to e has positive slackness ($\geq 3/4$). By complementary slackness conditions (and optimality of π) this implies that *every* optimal primal solution y satisfies $y_{uv} = 0$. Now, consider a removed pair $e = uv$, and let $w \in B - \{u, v\}$ be such that neither uw nor vw are removed (such a w exists by Lemma 5.14, Lemma 5.12, and our assumption on $|B|$). Then (P) has a constraint associated to uvw, which, for an optimal solution, reads $y_{uv} + y_{vw} + y_{uw} = y_{uv} \geq 1$. By optimality again, we infer $y_{uv} = 1$. Therefore y' is the *unique* optimal solution of (P), as desired. □

Proof. (Of Theorem 5.1 (2) when p is small) By Lemma 5.8,

$$Cost(Output) \leq OPT + d(\mathcal{A}, \widehat{E}).$$

By Lemma 5.18, we know that the optimal solution X^* of the semi-definite program matches $\mathcal{B}$ precisely on all vertex pairs incident to at least one cluster of size at least $c_4(np + \log n)$. By the design of the rounding algorithm, this implies that $\mathcal{A}$ also matches $\mathcal{B}$ precisely on for those clusters;

moreover, X^* is also optimal in the subgraph induced by the vertices in the remaining clusters. So we can ignore the large clusters and assume that all clusters have size at most $k = c_3(np + \log n)$.

Consider the clustering $\mathcal{B}'$ obtained from $\mathcal{B}$ by splitting clusters of size 2 in two. By Theorem 5.4,

$$\mathbf{E}_{alg}\left[d(\mathcal{A}, \widehat{E})\right] \leq 2.5\ d(\mathcal{B}', \widehat{E}).$$

We now proceed to compare $d(\mathcal{B}', \widehat{E})$ to OPT.

First, consider the case $np \leq c_2$, where c_2 is the constant from Theorem 5.9. We will bound the expected value, over the noisy process, of $d(\mathcal{B}', \widehat{E})$ and use Markov's inequality. Clearly a vertex pair u, v only contributes to $d(\mathcal{B}', \widehat{E})$ if one of the following three conditions holds.

- $B'_{uv} = 0$ and $E_{uv} = \widehat{E}_{uv} = 1$. We bound the expected number of such pairs by three times the expected number of unnatural triangles, i.e. $O((np)^3 + (np)^2 k)$ by Lemma 5.15.
- $B'_{uv} = 1$ and $E_{uv} = \widehat{E}_{uv} = 0$. We bound the expected number of such pairs by $O(n_3 kp)$ trivially, where n_3 is the number of vertices in base clusters of size at least 3 and $k = c_3(np + \log n)$ is the upper-bound on cluster size.
- $B'_{uv} = 1$, $E_{uv} = 1$ and $\widehat{E}_{uv} = 0$. We bound the number of such pairs by $O(n_3 p)$ using Lemma 5.13.

Altogether,

$$\begin{aligned} &\mathbf{E}_{graph}\left[d(\mathcal{B}', \widehat{E})\right] \\ &= O\left([(np)^3 + (np)^2 k] + [n_3 kp] + [n_3 p]\right) \\ &= \widetilde{O}((np)^2 + n_3 p) \end{aligned} \tag{5.2}$$

since $k = O(\log n)$ in this case. By Theorem 5.9 we have $OPT_{full} = \Omega(n_2 np + n^3 p^2)$ w.h.p.[7] and so $\mathbf{E}\left[d(\mathcal{B}', \widehat{E})\right] = \widetilde{O}(OPT_{full}/n)$, hence by Markov's inequality, with high probability we have $d(\mathcal{B}', \widehat{E}) = O(n^{-1/6})OPT_{full}$.

Second, consider the case $np > c_2$. Then $k = \widetilde{O}(np)$. Lemmas 5.15, 5.11 and 5.13 allow us to redo the proof of (5.2), replacing expectations by high probability statements, to show that

$$d(\mathcal{B}', \widehat{E}) = O\left([(np)^3 + (np)^2 k] + [n_3 kp] + [n_3 p] + \log^6 n\right) \tag{5.3}$$

with high probability. Now, (5.3) simplifies to $d(\mathcal{B}', \widehat{E}) = \widetilde{O}((np)^3) \leq \widetilde{O}(n^2 p) np^2 \leq \widetilde{O}(n^{-1/3})OPT_{full}$ w.h.p. using Theorem 5.9. Together, the two cases completes the proof. □

5.3.6 Proof of Theorem 5.1 (2) when $p \geq n^{-2/3}$

Define $M \bullet N \equiv \sum_{u,v} M_{uv} N_{uv}$, and for any $\{0,1\}$ matrix M, let

$$\widetilde{M}_{uv} = \begin{cases} -1 & \text{if } M_{uv} = 1 \\ 1 & \text{if } M_{uv} = 0. \end{cases}$$

[7] If $OPT_{full} = 0$ the present part 2 of Theorem 5.1 follows from part 1.

This gives a way to rewrite the objective of our semi-definite program. The following Lemma is trivial.

Lemma 5.19. *For any symmetric matrices M and N with $M_{uv} \in \{0,1\}$ and $0 \leq N_{uv} \leq 1$, we have*

$$d(M,N) = (1/2)(\widetilde{M} \bullet N - \widetilde{M} \bullet M)$$

The next Lemma is key. It is eventually used for $X = X^*$, the optimal solution to the SDP.

Lemma 5.20. *With high probability over the noisy model, the following holds: for every positive semi-definite matrix X with trace n,*

$$|\widetilde{E}_{full} \bullet X - \mathbf{E}\left[\widetilde{E}_{full}\right] \bullet X| \leq 5\sqrt{p}n^{3/2}.$$

Proof. Let $M = \widetilde{E}_{full} - \mathbf{E}\left[\widetilde{E}_{full}\right]$. Since X is symmetric, we can write $X = \sum_{i=1}^{n} \lambda_i v_i v_i^T$ where λ_i are the eigenvalues of X and v_i are corresponding unit-length eigenvectors. By elementary linear algebra, (using the fact that $Tr(AB) = Tr(BA)$ for two (m,n) and (n,m) matrices),

$$\begin{aligned} M \bullet X &= Tr(M^T X) = \sum_i Tr(M^T \lambda_i v_i v_i^T) \\ &= \sum_i \lambda_i Tr(v_i^T M v_i) = \sum_i \lambda_i v_i^T M v_i. \end{aligned}$$

Since X is positive semi-definite, the λ_is are non-negative and we get

$$\begin{aligned} |M \bullet X| &\leq \sum_i \lambda_i |v_i^T M v_i| \leq \sum_i \lambda_i \rho(M) \\ &= Tr(X)\rho(M) = n\rho(M), \end{aligned}$$

where $\rho(M)$ is the spectral radius of M. By Füredi and Komlós [1981] we have $\rho(M) \leq 5\sqrt{np}$, hence the Lemma. □

Proof. (Of Part 2 of Theorem 5.1 when $p \geq n^{-2/3}$) Since the maximum matching of U can only improve the cost, the cost of the output is at most $d(\mathcal{A}, E)$, where $\mathcal{A}$ denotes the output of SDP-Cluster. By the triangle inequality and Theorem 5.5,

$$\begin{aligned} \mathbf{E}_{alg}\left[d(\mathcal{A}, E)\right] &\leq \mathbf{E}_{alg}\left[d(\mathcal{A}, X^*)\right] + d(X^*, E) \\ &\leq 3d(\mathcal{B}, X^*) + d(X^*, E). \end{aligned}$$

Let $\mathcal{C}$ be an optimal clustering satisfying the conditions of Lemma 5.7. Since the semi-definite program is a relaxation, $d(X^*, \widehat{E}) \leq d(\mathcal{C}, \widehat{E})$. Lemma 5.7 implies that

$$d(X^*, E) \leq d(\mathcal{C}, E) + n/2 = OPT + O(n). \tag{5.4}$$

To see this, transform $\widehat{E}$ into E by switching vertex pairs one at a time. Each time vertex pair uv is switched, either u, v is a cluster of size 2 in $\mathcal{C}$ or the cost of $\mathcal{C}$ increases by 1. The cost of X^*, as that of any fractional clustering, increases by at most 1, hence Equation (5.4).

By Lemma 5.19, $d(\mathcal{B}, X^*) = (1/2)(\widetilde{\mathcal{B}} \bullet X^* - \widetilde{\mathcal{B}} \bullet \mathcal{B})$. In the fully random model, it is straightforward that $\mathbf{E}\left[\widetilde{E}_{full}\right] = (1-2p)\widetilde{\mathcal{B}}$, and so

$$(\widetilde{\mathcal{B}} \bullet X^* - \widetilde{\mathcal{B}} \bullet \mathcal{B}) = \frac{1}{1-2p}(\mathbf{E}\left[\widetilde{E}_{full}\right] \bullet X^* - \mathbf{E}\left[\widetilde{E}_{full}\right] \bullet \mathcal{B}).$$

Applying Lemma 5.20 to both $X = X^*$ and $X = \mathcal{B}$, we can write

$$d(\mathcal{B}, X^*) \leq \frac{1}{2(1-2p)}(\widetilde{E}_{full} \bullet X^* - \widetilde{E}_{full} \bullet \mathcal{B} - 10\sqrt{p}n^{3/2}).$$

Applying Lemma 5.19 again, once to $\widetilde{E}_{full} \bullet X^*$ and once to $\widetilde{E}_{full} \bullet \mathcal{B}$, we get

$$\widetilde{E}_{full} \bullet X^* - \widetilde{E}_{full} \bullet \mathcal{B} = 2(d(E_{full}, X^*) - d(E_{full}, \mathcal{B})).$$

We observe that

$$d(X^*, E_{full}) - d(\mathcal{B}, E_{full}) \leq d(X^*, E) - d(\mathcal{B}, E). \tag{5.5}$$

To see that, transform E_{full} into E by switching vertex pairs one at a time. Each time the adversary is "nice" (taking advantage of the semi-random model) and declines to add noise to vertex pair uv, the value of $\mathcal{B}$ decreases by 1 whereas the value of X^*, as that of any fractional clustering, decreases by at most 1, hence Equation (5.5).

We further claim that

$$d(X^*, E) - d(\mathcal{B}, E) \leq d(X^*, \widehat{E}) - d(\mathcal{B}, \widehat{E}) + \tilde{O}(np+1). \tag{5.6}$$

To see that, we transform E into $\widehat{E}$ by switching vertex pairs one at a time. Each time the algorithm removes an edge of E that are between different clusters of $\mathcal{B}$, the value of $\mathcal{B}$ decreases by 1 whereas the value of X^*, as that of any fractional clustering, decreases by at most 1. Each time the algorithm removes an edge of E that is inside a cluster of $\mathcal{B}$, the value of $\mathcal{B}$ changes by 1 and the value of X^*, as that of any fractional clustering, changes by at most 1, so the difference changes by at most 2. Hence the difference from $d(X^*, E) - d(\mathcal{B}, E)$ to $d(X^*, \widehat{E}) - d(\mathcal{B}, \widehat{E})$ is at most twice the number of edges inside clusters of $\mathcal{B}$ that are removed by the algorithm. By Lemma 5.13 there are $\tilde{O}(np+1)$ such pairs, hence Equation (5.6).

By optimality of X^*, we have $d(\widehat{E}, X^*) - d(\widehat{E}, \mathcal{B}) \leq 0$. Clearly $O(n) + \tilde{O}(np+1) = O(\sqrt{p}n^{3/2})$ for $p \geq n^{-2/3}$. Altogether this yields

$$Cost(OUT) \quad \leq \quad OPT + O\left(\frac{\sqrt{p}n^{3/2}}{1-2p}\right).$$

By Theorem 5.9, $OPT_{full} = \Omega(n^2 p)$, and so

$$(Cost(OUT) - OPT)/OPT_{full} = O\left(\frac{1}{\sqrt{np}(1/2-p)}\right).$$

It is easy to check that for $n^{-2/3} \leq p \leq 1/2 - n^{-1/3}$, this quantity is $O(n^{-1/6})$. □

5.3.7 Proof of Theorem 5.1 (3)

Part (3) of Theorem 5.1 is proved using the same techniques as used to prove part (2). When p is large we choose lower bound $L = d(X^*, E) - n/2$, which is valid by Equation (5.4) and was implicitly shown to be sufficiently tight in the last section.

When p is small the argument is a bit more involved. First note that the proof of Lemma 5.18 can be readily converted into a polynomial-time algorithm for certifying that a particular cluster is in every optimal clustering. This algorithm succeeds with high probability on any base cluster B of size at least $c_3(np + \log n)$. We run this algorithm (solve an LP and check complementary slackness) on every output cluster $A \in \mathit{Output}$. We discard those clusters that this algorithm successfully certifies and compute initial lower bound L_1 equal to what Output (and hence $\mathcal{C}$) pays for the edges incident to the discarded clusters. Lemma 5.8 implies that$L_2 = d'(\mathit{Output}, E) - d'(\mathcal{A}, \widehat{E})$ is a lower bound on the instanced induced by the remaining vertices, where $d'(\cdot, \cdot)$ is like $d(\cdot, \cdot)$ but is limited to the subgraph induced by the remaining vertices. Our overall lower bound is $L = L_1 + L_2$.

5.4 Proof of Theorem 5.2

This proof is inspired by the analysis in Feige and Krauthgamer [2000] for the planted clique problem. Our key innovation is extending the results of Füredi and Komlós Füredi and Komlós [1981] to random matrices with entries only partially independent (Lemma 5.26).

5.4.1 Overall Proof

First we give a brief outline of our proof. First we define a new SDP (5.7) which is (up to scaling) a relaxation of the SDP in our SDPCluster algorithm. Then we write its dual (5.8). We define a dual solution (Π, Υ) in (5.9). Finally we analyze eigenvalues (Lemmas 5.21-5.24) to prove that our dual solution is feasible. Finally we use the fact that our dual feasible solution has the same objective value as $\mathcal{B}$ in the primal, plus a few more arguments, to prove the theorem.

By Lemma 5.12 with high probability all edges within base clusters remain in $\widehat{E}$. Edges between clusters that are not in $\widehat{E}$ can be accounted for as semi-randomness, so we can ignore the distinction between $\widehat{E}$ and E. Semi-randomness can be dealt with easily (using Equation (5.5) in Section 5.3.6) so we focus on the fully random case.

Throughout this section we assume that there are b clusters of size at least $\gamma \geq c_1\sqrt{n}$ for some constant c_1 to be decided later. Let $\mu = 1 - 2p \geq 1/3$.

Let J^{uv} denote a matrix with u, v entry equal to 1 and all others 0. Following Alizadeh [1995], we write $M \succeq N$ if $M - N$ is positive semi-definite. Recall the definition $\widetilde{E}_{uv} = \begin{cases} -1 & \text{if } E_{uv} = 1 \\ 1 & \text{if } E_{uv} = 0 \end{cases}$.

Observe that X^* is feasible in the following SDP:

$$\max_X \ -\widetilde{E} \bullet X \ \text{ s.t.} \tag{5.7}$$
$$\begin{cases} J^{uu} \bullet X = 1 & \forall u \in V \quad \text{(Dual variable } \Pi_u\text{)} \\ -J^{uv} \bullet X \leq 0 & \forall u, v \in V \quad \text{(Dual variable } -\Upsilon_{uv}\text{)} \\ X \succeq 0 & \text{(positive semi-definite)} \end{cases}$$

Recall from Lemma 5.19 that $d(E, X) = (1/2)(\widetilde{E} \bullet X - \widetilde{E} \bullet E)$ so up to rescaling SDP (5.7) relaxes the SDP in SDPCluster (by eliminating the triangle inequalities).

For convenience we package the dual variables Π_u and $-\Upsilon_{uv}$ (the minus sign will be convenient later) into diagonal matrix Π and symmetric matrix $-\Upsilon$. Here is the dual of SDP (5.7):

$$\min_{\Pi, \Upsilon} \ \sum_v \Pi_v \ \text{ s.t.} \tag{5.8}$$
$$\begin{cases} \Pi - (-\Upsilon) \succeq -\widetilde{E} \\ \Pi \text{ diagonal} \\ -\Upsilon \geq 0 \qquad \text{(All entries non-negative)} \end{cases}$$

We rewrite the first constraint in the dual as $M \equiv \widetilde{E} + \Pi + \Upsilon \succeq 0$.

Recall that the vertex set V is partitioned into b base clusters $B_1, \ldots, B_b$. For this proof we subdivide[8] the clusters into *subclusters* $S_1, \ldots, S_r$ so that $\gamma/2 \leq |S_i| \leq \gamma$ for all i. For vertex u let $B(u)$ and $S(u)$ denote the cluster and subcluster u is in respectively. We now present our dual solution (Π, Υ). We choose

$$\Pi_{uu} = -\sum_{v \in B(u)} \widetilde{E}_{uv}. \tag{5.9}$$

Let Υ_{uv} be zero when $B(u) = B(v)$ and

$$\sum_{i \in S(u), j \in S(v)} \frac{\widetilde{E}_{ij}}{|S(u)||S(v)|} - \sum_{i \in S(u)} \frac{\widetilde{E}_{iv}}{|S(u)|} - \sum_{j \in S(v)} \frac{\widetilde{E}_{uj}}{|S(v)|}$$

otherwise. This dual solution satisfies two key (and easily verified) properties:

1. The vector with all components in one base cluster equal to 1 and all others zero is an eigenvector of M with eigenvalue 0.
2. The dual objective value $\sum_v \Pi_v$ equals $-\widetilde{E} \bullet \mathcal{B}$, the primal value of $\mathcal{B}$.

Let us first show that $-\Upsilon_{uv} > 0$ with high probability for any u, v with $B(u) \neq B(v)$. Observe that each of the three sums in the definition of Υ_{uv} are an average of $\Omega(\gamma)$ independent $-1/1$ random variables each with mean μ. Each of the three sums therefore equals its expectation, namely μ, plus or minus $O\left(\sqrt{\frac{\log n}{\gamma}}\right) < \mu/10$ by the Azuma-Hoeffding inequality. We conclude that

$$\Upsilon_{uv} < 0. \tag{5.10}$$

[8] On first reading consider the special case of all clusters having size exactly γ and $B_i = S_i$.

As noted above M has b orthogonal eigenvectors with eigenvalue 0. We next show that the $b+1$ smallest eigenvalue of M is strictly positive, which will imply that M is positive semi-definite. To do so we decompose M as

$$\begin{aligned} M &= \widetilde{E} + \Pi + \Upsilon \\ &= \Pi + \underbrace{\left(\mathbf{E}\left[\widetilde{E}+\Upsilon\right]\right)}_{\equiv M^{(1)}} + \underbrace{\left(\widetilde{E}-\mathbf{E}\left[\widetilde{E}\right]\right)}_{\equiv M^{(2)}} + \underbrace{\left(\Upsilon-\mathbf{E}\left[\Upsilon\right]\right)}_{\equiv M^{(3)}} \end{aligned}$$

and analyze the eigenvalues of Π, $M^{(1)}$, $M^{(2)}$ and $M^{(3)}$ separately. In particular we will prove the following four Lemmas:

Lemma 5.21. *All eigenvalues of* Π *are* $\Omega(\sqrt{\gamma})$ *with high probability.*

Lemma 5.22. *The* $b+1$ *smallest eigenvalue of* $M^{(1)}$ *is* μ*.*

Lemma 5.23. $M^{(2)}$ *has all eigenvalues at least* $-\Theta(\sqrt{n})$ *with high probability.*

Lemma 5.24. $M^{(3)}$ *has all eigenvalues at least* $-\Theta(\sqrt{n})$ *with high probability.*

Before proving these Lemmas we show how they imply Theorem 5.2. We use the following corollary of the Courant-Fischer Theorem, due to Weyl:

Theorem 5.25 (Weyl Horn and Johnson [1985])**.** *For any* $n \times n$ *symmetric matrices* A *and* B *and integer* $1 \leq k \leq n$ *the* k*th smallest eigenvalue of* $A+B$ *is at least the* k*th smallest eigenvalue of* A *plus the smallest eigenvalue of* B*.*

We apply Theorem 5.25 three times with $k = b+1$ together with Lemmas 5.21, 5.22, 5.23 and 5.24. This shows that the $b+1$ smallest eigenvalue of M is at least $\Omega(\sqrt{\gamma}) - \mu - \Theta(\sqrt{n}) - \Theta(\sqrt{n}) > 0$ for sufficiently large $\gamma = c_1\sqrt{n}$. We conclude $M \succeq 0$ and hence our dual assignment is feasible. As already remarked this dual assignment has the same objective value as $\mathcal{B}$ in the primal, so we conclude that both are optimal.

We will now prove Theorem 5.2 by showing that $\mathcal{B}$ is not only *a* primal optimal but the *unique* optimum.

Proof. (Of Theorem 5.2) Let X^* be an arbitrary optimal solution of the SDPCluster SDP. First consider some u, v in different clusters. We previously showed (5.10) that dual variable $-\Upsilon_{uv}$ is strictly positive, hence by the complementary slackness condition $X^*_{uv} \cdot -\Upsilon_{uv} = 0$ we conclude $X^*_{uv} = 0$.

For u, v in different clusters we use the complementary slackness condition $MX^* = 0$ Alizadeh [1995]. This implies that any eigenvector of X^* with non-zero eigenvalue must be in the nullspace of M. In the previous paragraph we showed that X^* is block diagonal with one block per cluster so w.l.o.g. assume eigenvectors of X^* each have their support contained within a single cluster. Our previous analysis of the eigenspectrum of M implies that the nullspace of M is spanned by vectors $x_1, x_2, \ldots, x_b$ where x_i has components equal to 1 within base cluster B_i and 0 elsewhere. We can therefore write X^* as a linear combination of rank-one outer product matrices $x_i x_i^T$. The constraint $X_{uu} = 1$ imply that the linear combination has coefficients all 1, i.e. $X^* = \mathcal{B}$. ∎

5.4.2 Eigenvalue analysis

The following Lemma, proved shortly in Section 5.4.3, will be helpful for proving Lemmas 5.23 and 5.24. Füredi and Komlós Füredi and Komlós [1981] showed that a symmetric matrix with independent random entries, each with variance σ^2, has spectral radius $(2+o(1))\sigma\sqrt{n}$. Our Lemma generalizes their result to matrices whose entries are not completely independent, but independent outside certain roughly equal sized blocks.

Lemma 5.26 (Generalization of Füredi and Komlós [1981])**.** *Let set V, $|V| = n \geq 64$, be partitioned into r* classes *$S_1 \dots S_r$, all of size between $\beta/2$ and β for some $1 \leq \beta \leq n$. Let $S(v)$ denote the class that $v \in V$ is in. Let N be a matrix-valued random variable indexed by V where:*

- *$N_{uv} = N_{vu}$ for all $u, v \in V$ (symmetric)*
- *$\mathbf{E}[N] = 0$*
- *The blocks of matrix N induced by the class structure are mutually independent. The block indexed by $1 \leq i, j \leq r$ is the set of all random variables N_{uv} with $\{S(u), S(v)\} = \{i, j\}$.*
- *$\mathbf{E}\left[N_{uv}^2\right] \leq \sigma^2$ and $|N_{uv}| \leq K$ for all $u, v \in V$ and some uniform K and σ with $K \geq \sigma > 0$.*
- *$\beta \cdot \frac{K^2}{\sigma^2} \leq n/(800 \lg^4 n) = \widetilde{O}(n)$*

Then with probability at least $1 - n^{-8}$ all eigenvalues of N have absolute value at most $20\sigma\sqrt{\beta n}$, i.e. the spectral radius of N is $O(\sigma\sqrt{\beta n})$.

Lemma 5.26 with $\beta = 1$ is equivalent up to constants to Füredi and Komlós [1981]. We use Lemma 5.26 twice, once with $\beta = 1$ and once with β equal to the cluster size lower-bound γ.

Proof. (Of Lemma 5.21) Matrix Π is diagonal so its eigenvalues are simply the entries on its diagonal. Consider entry Π_u for vertex u in cluster $B(u)$. Clearly Π_u consists of the sum of $|B(u)| - 1$ independent random variables each with mean μ. Therefore by the Azuma-Hoeffding inequality we have $\Pi_u = \mu(|B(u)| - 1) \pm O(\sqrt{|B(u) - 1| \log n}) = \Omega(\gamma)$ with high probability. A union bound over vertices completes the proof. □

Proof. Of Lemma 5.22) Note that $\mathbf{E}[\Upsilon_{ij}] = -\mathbf{E}\left[\widetilde{E}_{ij}\right]$ for all i and j in different clusters hence $M^{(1)}$ is block diagonal with one block per cluster. Consider some block N corresponding to cluster B_i. Clearly $N = -\mu J + \mu I$, hence N has eigenvalue $-\mu|B_i| + \mu$ with multiplicity 1 and eigenvalue μ with multiplicity $|B_i| - 1$. Unioning these spectra over the clusters proves the Lemma. □

Proof. (Of Lemma 5.23) Observe that the entries of $M^{(2)}$ are independent, have variance $O(p)$ and are bounded by 2. The result of Füredi and Komlós, i.e. Lemma 5.26 with $s = n$ and $\beta = 1$, proves the Lemma. □

Proof. (Of Lemma 5.24) It is easy to verify that $M^{(3)}$ satisfies the conditions of Lemma 5.26 with $\beta = \gamma$, $\sigma = \Theta(\frac{1}{\sqrt{\gamma}})$ and $K = \Theta\left(\sqrt{\frac{\log n}{\gamma}}\right)$, the last by Azuma-Hoeffding inequality. Applying Lemma 5.26 proves the Lemma. □

5.4.3 Proof of Lemma 5.26

We extend the techniques of Füredi and Komlós Füredi and Komlós [1981]. Let λ_i denote the ith eigenvalue of matrix N. Let $k = 10\lceil \lg n \rceil$, an even integer. Clearly

$$\begin{aligned}
&\Pr\left(\max_i |\lambda_i| \geq 20\sigma\sqrt{\beta n}\right) \\
&\leq \Pr\left(\sum_i \lambda_i^k \geq (20\sigma\sqrt{\beta n})^k\right) \\
&\leq \mathbf{E}\left[\sum_i \lambda_i^k\right] / (20\sigma\sqrt{\beta n})^k \qquad \text{(Markov)}. \qquad (5.11)
\end{aligned}$$

The sum of λ_i^k is the trace of N^k, which can alternatively be written as $\sum_q \prod_{e\in q} N_e$, where the sum is over walks q of length k whose ending point equals their starting point. Consider such a walk q.

If there are two classes S_i and S_j such that q has exactly one edge f between S_i and S_j, then

$$\mathbf{E}\left[\prod_{e\in q} N_e\right] = \mathbf{E}\left[N_f\right]\mathbf{E}\left[\prod_{e\in q, e\neq f} N_e\right] = 0$$

by independence and since $\mathbf{E}[N_f] = 0$ by assumption.

If not, let p denote the number of classes visited by walk q. Let S_i and S_j be two of them and assume that q has $\ell \geq 2$ visits of edges between them. Then one can check (using the inequalities of Hölder and Cauchy-Schwarz) that $\mathbf{E}\left[\prod_{e\in q, e\in S_i\times S_j} N_e\right] \leq \sigma^2 K^{\ell-2}$. By independence,[9] we deduce

$$\mathbf{E}\left[\prod_{e\in q} N_e\right] \leq \sigma^{2(p-1)} K^{k-2(p-1)}$$

We group walks q of this type by the corresponding walk q' over the classes, where walk q over V *refines* walk q' over the set of classes if the class of each point in q is equal to the corresponding point in q'. Any such q' must clearly use each *superedge* between classes at least twice or not at all. For any fixed q' there are at most β^k walks q that refine q'. Therefore for any q'

$$\mathbf{E}\left[\sum_{q \text{ refines } q'} \prod_{e\in q} N_e\right] \leq \beta^k \sigma^{2(p-1)} K^{k-2(p-1)} \qquad (5.12)$$

We now use the following Lemma implicit in Füredi and Komlós [1981]:

Lemma 5.27 (Füredi and Komlós [1981])**.** *The number of walks (cyclic or not) of k steps on the complete graph with r vertices that visit exactly p distinct vertices and visit each non-loop edge at least twice (in either direction) is at most*

$$r\cdot(r-1)\cdot\ldots\cdot(r-p+1)\cdot\binom{k}{k-2p+2}p^{2(k-2p+2)}\frac{1}{p}\binom{2p-2}{p-1}.$$

[9] And the assumption $\sigma \leq K$.

Let $T(p)$ denote the contribution of walks that visit p distinct classes to the expected trace of N^k. Combining Lemmas 5.27 and (5.12) we see that for any $p \le k/2+1$.

$$\begin{aligned} & T(p) \le && (5.13)\\ & r^p \binom{k}{k-2p+2} p^{2(k-2p+2)} \frac{1}{p} \binom{2p-2}{p-1} \cdot \beta^k \sigma^{2p-2} K^{k-2p+2} \\ & \le r^p 2^k p^{2(k-2p+2)} \cdot 1 \cdot 2^{2p-2} \cdot \beta^k \sigma^{2p-2} K^{k-2p+2} \\ & = \left(2p^2 \beta K\right)^k \left(\frac{4\sigma^2 r}{p^4 K^2}\right)^{p-1} r && (5.14)\\ & \equiv \bar{T}(p) \end{aligned}$$

Recall that $k = 10\lceil \lg n \rceil$ hence $p \le \frac{k}{2}+1 \le 5\lceil \lg n \rceil + 1 \le 6 \lg n$. Also recall the assumption that $\beta \cdot \frac{K^2}{\sigma^2} \le n/(400 \lg^4 n)$, hence $\frac{\sigma^2 n}{(\lg n)^4 K^2 \beta} \ge 400$. Finally note the trivial fact $r \ge n/\beta$. Putting these facts together we conclude

$$\left(\frac{4\sigma^2 r}{p^4 K^2}\right) \ge \left(\frac{4\sigma^2 n}{(6 \lg n)^4 K^2 \beta}\right) \ge \frac{4 \cdot 800}{6^4} > 2. \tag{5.15}$$

Combining (5.14) and (5.15) we conclude

$$\begin{aligned} \mathbf{E}\left[\sum_i \lambda_i^k\right] = \sum_{p=1}^{r} T(p) &\le \sum_{p=1}^{k/2+1} \bar{T}(p) \\ &\le 2\bar{T}(k/2+1) = 2\left(4\beta\sigma\sqrt{r}\right)^k \cdot 2r. && (5.16) \end{aligned}$$

We now finish the proof of Lemma 5.26:

$$\begin{aligned} & \Pr\left(\max_i |\lambda_i| \ge 20\sigma\sqrt{\beta n}\right) \\ & \le \mathbf{E}\left[\sum_i \lambda_i^k\right] / (20\sigma\sqrt{\beta n})^k && \text{(by 5.11)} \\ & \le \frac{\left(4\beta\sigma\sqrt{r}\right)^k 2r}{(20\sigma\sqrt{\beta n})^k} && \text{(by 5.16)} \\ & \le \left(\frac{1}{2}\right)^k \cdot 2n && (r \le 2n/\beta \text{ and } r \le n) \\ & \le n^{-8} && (k = 10\lceil \lg n \rceil). \end{aligned}$$

5.5 Proof of Theorem 5.3

5.5.1 Preliminaries

Algorithm 5.3 proceeds by first identifying a candidate set of clusters $\mathcal{A}$ and then certifying that the large clusters therein, denoted $\mathcal{A}'$, are in fact optimal. The following four Lemmas easily imply Theorem 5.3.

Lemma 5.28. *W.h.p. Algorithm 5.3 produces intermediate set of clusters $\mathcal{A}$ that includes all clusters of the base clustering $\mathcal{B}$ that have size greater than or equal to $3\lceil 70/\delta \rceil$.*

Lemma 5.29. *W.h.p. Algorithm 5.3 produces intermediate set of clusters $\mathcal{A}$ that includes no clusters except those guaranteed by Lemma 5.28.*

Lemma 5.30. *Conditioned on the w.h.p. events of Lemmas 5.28 and 5.29 occurring Algorithm 5.3 outputs "failure" with probability at most $2n^{-5}$.*

Lemma 5.31. *Let $\mathcal{A}'$ be a clustering output by Algorithm 5.3 and $\mathcal{C}$ be an optimal clustering. We have $\mathcal{A}' = \{ C \in \mathcal{C} \ : \ |C| \geq 15s' \}$.*

We prove each of these Lemmas in turn. Throughout this section we assume that $p \leq n^{-\delta}/60$ for some $\delta > 0$ and $s = \lceil 70/\delta \rceil$.

For edge set E and disjoint vertex sets S and T, let $E(S,T)$ denote the number of edges with one endpoint in S and the other in T. For vertex v let $E(v,T)$ denote $E(\{v\}, T \setminus \{v\})$.

Recall that $s \geq \lceil 70/\delta \rceil$. If $p \leq n^{-5}$, then with probability $1 - n^{-3}$ the input graph is just the base clustering with no noise, in which case cluster-finding is trivial. We can therefore safely assume $\delta < 5$, and hence $s \geq 40/\delta + 5$.

A *mistake* is an edge whose label is different in the base clustering and in the input clustering.

Lemma 5.32. *Let X be a set of edges of cardinality at least $70/\delta$. Then, with probability at least $1 - n^{-7}$, there are at most $|X|/10$ mistakes in X.*

Proof. Elementary counting: $\binom{|X|}{|X|/10} p^{|X|/10} \leq \left(\frac{e|X|p}{|X|/10} \right)^{|X|/10} \leq n^{-\delta |X|/10} \leq n^{-7}$. □

Lemma 5.33. *With probability at least $1 - n^{-5}$, every subgraph induced by a vertex set Y of size $k \geq 40/\delta + 5$ has at most $|Y|^2/40$ mistakes.*

Proof. Let $k \geq 40/\delta + 5$ be given. There are $\binom{n}{k} \leq n^k$ subsets of size k. Each subset has $\binom{k}{2} \leq k^2/2$ vertex pairs within a given subset, so the mean number of mistakes is at most $k^2 p/2$. Using Markov and Chernoff bounds, write:

$$\Pr\left(\exists S \text{ of size } k \text{ with at least } k \text{ mistakes}\right) \leq n^k \left(\frac{epk^2/2}{k^2/40} \right)^{k^2/40} \leq n^{k - \delta k^2/40}$$

We therefore want $k - \delta k^2/40 \leq -5$. By Taylor series around $k = 40/\delta$, we see that $k - \delta k^2/40 \leq -(k - 40/\delta)$, hence $k \geq 40/\delta + 5$ suffices. □

Lemma 5.34. *Let U, W be disjoint vertex sets and $s > 0$ an integer. Assume that for every subset S of U and T of W with cardinalities $|S| = |T| = s$, there are at most (resp. at least) $(.35)|S||T|$ edges between S and T. Then for every subset S' of U and T' of W with cardinalities $|S'|, |T'| \geq s$, there are also at most (resp. at least) $(.35)|S'||T'|$ edges between S' and T'.*

Proof. Here is a way to compute the number of edges between S' and T' divided by $|S'||T'|$: Pick a random subset S of S' of size s and a random subset T of T' also of size s, compute the number of edges between S and T divided by s^2, and average over the random choices of S, T. The number of edges between S and T is bounded by assumption, hence the lemma. □

For every cluster $B \in \mathcal{B}$ of size at least $3s$, fix an arbitrary seed set $K_B \subset B$ of size s.
Let *Event 1, ... Event 4* refer to:

1. For all $v \in V$ and $B \in \mathcal{B}$ with $|B| \geq 3s$, $E(v, K_B) \geq |K_B|/2$ if and only if $v \in B$.
2. For all $v \in V$ and $B \in \mathcal{B}$ with $|B| \geq 3s$, $E(v, B) \geq .9|B|$ if $v \in B$ and $E(v, B) \leq .1|B|$ if $v \notin B$.
3. All $B \in \mathcal{B}$ and disjoint sets $T, S \subseteq B$ with $|T| = |U| = s$ satisfy $E(T, U) \geq .9s^2$
4. For every pair of disjoint vertex sets S, T with cardinalities $|S| = |T| = s$, such that no cluster $B \in \mathcal{B}$ intersects both S and T ($\forall B, B \cap S = \emptyset$ or $A \cap T = \emptyset$), there are at most $(.1)|S||T|$ edges between S and T.

Lemma 5.35. *Events 1,2,3,4 all occur with probability at least* $1 - O(n^{-5})$.

Proof. Events 1 and 2 occur with that probability by Lemma 5.32 and a union bound over the $O(n^2)$ possible combinations of base cluster B and vertex v. Events 3 and 4 follow from Lemma 5.33 using $X = S \cap T$. □

We henceforth assume that events $1, \ldots 4$ occur. We refer to the three bullets in the first half of Algorithm 5.3 as *Condition 1 ... 3*. We refer to the three bullets in the second half of Algorithm 5.3 as *Property 1 ... 3*.

5.5.2 Proof of Lemma 5.28

In this subsection we prove that every cluster $B \in \mathcal{B}$ of size at least $3s$ is added to $\mathcal{A}$ at some point.

We fix some cluster $B \in \mathcal{B}$ of size at least $3s$. Let K be its seed. We show that when Algorithm 5.3 considers $S = K$, then it adds $A = B$ to $\mathcal{A}$. Event 1 and the definition of A implies $A = B$. Events 3, 4 and 2 guarantee that A satisfies conditions 1, 2 and the second part of condition 3 respectively. The first part of condition 3 is satisfied by assumption on the size of B.

5.5.3 Proof of Lemma 5.29

In this subsection we show that every cluster A added to $\mathcal{A}$ satisfies $|A| \geq 3s$ and $A \in \mathcal{B}$.

Lemma 5.36. *For every cluster A that satisfies conditions 1–3 there exists $B \in \mathcal{B}$ such that $|B \backslash A| < s$ and $|A \setminus B| < s$.*

Proof. Construct a bipartition S, T of A as follows. Consider the clusters $B_i \in \mathcal{B}$ in descending order of $|B_i \cap A|$, adding $B_i \cap A$ to whichever of S, T have fewer vertices. It is well known that this greedy algorithm satisfies $||S| - |T|| \leq |B^+ \cap A|$, where B^+ is the cluster with the largest intersection

with A. Without less of generality suppose that $|T| \geq |S|$. If $|S| \geq s$, then $E(S,T) \leq .1|S||T|$ by event 4 and Lemma 5.34, contradicting condition 1, which implies $E(S,T) \geq .9|S||T|$, again using Lemma 5.34. Therefore $|S| < s$. Therefore $|T| < s + |B^+ \cap A|$, so $3s \leq |S| + |T| < 2s + |B^+ \cap A|$, so $|B^+ \cap A| > s$. Therefore the greedy procedure placed all other vertices in S, so $|A \setminus B^+| = |S| < s$.

By Condition 3 and $|A \setminus B^+| < s$ we see that $|A \cap B^+| \geq 2s$. If $|B^+ \setminus A|$ were s or more, we would detect that because then $E(A \cap B^+, B^+ \setminus A) \geq .9|A \cap B^+||B^+ \setminus A|$ by event 3, which contradicts condition 2. □

Every A that passes is approximately equal to some $B \in \mathcal{B}$ by Lemma 5.36. We now show that $A = B$.

First consider some vertex $v \in B$. We use Event 2 and properties 3 to write $E(v,A) \geq E(v, A \cap B) \geq |A \cap B| - .1|B| \geq (|A| - s) - .1(|A| + s) = .9|A| - 1.1s \geq .9|A| - \frac{1.1}{3}|A| > |A|/2$, so $B \subseteq A$ by condition 3. For $v \notin B$, we similarly write $E(v,A) = E(v, A \cap B) + E(v, A \setminus B) \leq .1|B| + s \leq .1(|A| + s) + s = .1|A| + 1.1s \leq .1|A| + \frac{1.1}{3}|A| < |A|/2$. Therefore $A = B$.

5.5.4 Proof of Lemma 5.30

The proofs of the first property and of the second property are identical: for each v and A we apply Lemma 5.32 to the set of edges between v and A, and use the union bound. (There are at most n^2 such sets.)

To prove the third property, observe that each cluster $B_i \in \mathcal{B}$ which intersects both S and T has c_i vertices in S and c_i' vertices in T, with $c_i + c_i' \leq 3s$. Therefore in the base clustering the number of edges between S and T is $\sum_i c_i c_i' \leq \sum_i (c_i + c_i')^2/4$. By convexity and using $\sum_i (c_i + c_i') \leq |S \cup T| = 12s$ and $\max_i (c_i + c_i') \leq 3s$, this is bounded by $36s^2/4 = |S||T|/4$. By Lemma 5.33 applied to $X = S \cup T$, the mistakes add another $(12s)^2/40 = |S||T|/10$ edges, for a total of at most $.35|S||T|$ edges.

5.5.5 Proof of Lemma 5.31

Let $M(S,T) = 2E(S,T) - |S||T|$, which is the profit from merging clusters S and T into a new cluster.

Lemma 5.37. *Let $\mathcal{A}$ be the input partial clustering that Algorithm 5.3 output and $\mathcal{C}$ be an optimal clustering. For any $C \in \mathcal{C}$ satisfying $|C| \geq 18s$ there exists a unique $A \in \mathcal{A}$ such that:*

1. $|C \setminus A| < 6s$
2. $|A| \geq 3|C|$ *or* $C \subseteq A$

Proof. Suppose $|C| \geq 18s$. Let A be the cluster in $\mathcal{A}$ maximizing $|A \cap C|$.

Relabel the clusters $A_i \in \mathcal{A}$ so that $|A_1 \cap C| \geq |A_i \cap C|$ for all $1 \leq i \leq |\mathcal{A}|$. Let $S_k = \bigcup_{i=1}^{k} A_i \cap C$ and let $S = S_k$ where k is the smallest integer such that $|S_k| \geq 6s$. (If no such k exists, let S be $\bigcup_i A_i \cap C$ plus sufficient additional vertices from $C \setminus \bigcup_i A_i \cap C$ so that $|S| = 6s$.) Let $T = C \setminus S$. If $|T|$ were at least $6s$ then Lemma 5.34 and Property 3 would contradict optimality of $\mathcal{C}$ so $|T| < 6s$.

We know $|S| + |T| \geq 18s$ so therefore $|S| > 12s$. The definition of k and the relabeling imply $k = 1$, proving the first statement of the Lemma.

For $v \in C \setminus A$ write $E(v, C \cap A) \leq E(v, A) \leq .1|A|$ using Property 2b. Note that $M(C \setminus A, C \cap A) \geq 0$ by optimality of C, so therefore $0 \leq M(C \setminus A, C \cap A) = \sum_{v \in C \setminus A} M(v, C \cap A) = \sum_v 2E(v, C \cap A) - |C \cap A| \leq |C \setminus A|(.2|A| - |C \cap A|)$. Suppose $C \not\subseteq A$, hence $|C \setminus A| > 0$. Therefore $0 \leq .2|A| - |C \cap A|$ hence using $|C| \geq 18s$ we see $|A| \geq 5|C \cap A| \geq \frac{10}{3}|C|$, proving the second statement of the Lemma. □

We are now ready to prove Lemma 5.31. We first show that $\mathcal{A}' \subseteq \{ C \in \mathcal{C} \ : \ |C| \geq 45s \}$.

Let A be the largest cluster in $\mathcal{A} \setminus \mathcal{C}$. For sake of contradiction suppose $|A| > 45s$. Consider the clustering $\mathcal{C}' = \{A\} \cup \{ C \setminus A \ : \ C \in \mathcal{C} \}$. We will prove that $\mathcal{C}'$ is strictly better than $\mathcal{C}$, contradicting the optimality of $\mathcal{C}$.

Let C_u denote the cluster of $\mathcal{C}$ containing vertex u. Let $C_1, C_2 \ldots C_l$ denote the clusters of $\mathcal{C}$ with non-empty intersection with A^*. For $\mathcal{C}'$ and $\mathcal{C}$, the objective function only differs for pairs $\{u, v\}$ such that $u \in C_u \cap A, v \in C_u \setminus A$ (in which case we charge the change to C_u), or such that $u \in C_u \cap A, v \in C_v \cap A$ with $C_u \neq C_v$ (in which case we charge half of the change to C_u and half to C_v). The change in objective function can thus be written as the sum, over $1 \leq i \leq l$, of $\Delta\text{profit}(i)$, where $\Delta\text{profit}(i)$ equals

$$-M(C_i \cap A, C_i \setminus A) + \frac{1}{2} M(C_i \cap A, A \setminus C_i).$$

Let D be the smaller of $C_i \cap A$ and $A \setminus C_i$, hence $|D| \leq (1/2)|A|$. Using Property 2a we see

$$\begin{aligned} E(D, A \setminus D) &= \sum_{v \in D} E(v, A \setminus D) \\ &\geq \sum_v E(v, A) - |D| \geq |D|(.9|A| - |D|). \end{aligned}$$

Therefore

$$\begin{aligned} M(D, A \setminus D) &= 2E(D, A \setminus D) - |D| \cdot |A \setminus D| \\ &\geq |D|(1.8|A| - 2|D| - |A \setminus D|) \\ &= |D|(.8|A| - |D|) \geq .3|D| \cdot |A| > 0 \end{aligned}$$

allowing us to write

$$\begin{aligned} &\Delta\text{profit}(i) \\ &\quad \geq -M(C_i \cap A, C_i \setminus A) + \min(|C_i \cap A|, |A \setminus C_i|) \cdot \\ &\qquad \cdot (.4|A| - (1/2)\min(|C_i \cap A|, |A \setminus C_i|)). \quad (5.17) \end{aligned}$$

We show that $\Delta\text{profit}(i) > 0$ for all i by cases on $|C_i \cap A|$.

Case 1: $|C_i| < .4|A|$. We trivially see $M(C_i \cap A, C_i \setminus A) \leq |C_i \cap A|(|C_i| - |C_i \cap A|)$. Using (5.17)

$$\begin{aligned}
&\Delta\text{profit}(i) \\
&\geq \quad -|C_i \cap A|(|C_i| - |C_i \cap A|) + \\
&\qquad + \frac{1}{2}|C_i \cap A|(.8|A| - |C_i \cap A|) \\
&= \quad |C_i \cap A| \left(-(|C_i| - |C_i \cap A|) + .4|A| - \frac{1}{2}|C_i \cap A| \right) \\
&\geq \quad |C_i \cap A| \left(.4|A| - |C_i| \right) > 0.
\end{aligned}$$

Case 2: $|C_i| \geq .4|A|$. We assumed $|A| \geq 45s$ hence $|C_i| \geq .4 \cdot 45s = 18s$. Let A' be the cluster promised by Lemma 5.37. Clearly $A' \notin \mathcal{C}$ hence $|C| \geq .4|A| \geq .4|A'| > \frac{1}{3}|A'|$ so by Lemma 5.37 we have $C_i \subseteq A$. Therefore $M(C_i \cap A, C_i \setminus A) = 0$ hence using (5.17) $\Delta\text{profit}(i) > 0$.

Thus in all cases the change in profit is positive: $\mathcal{C}'$ is strictly better than $\mathcal{C}$, contradicting the optimality of $\mathcal{C}$. This completes the proof that $\mathcal{A}' \subseteq \{ C \in \mathcal{C} \ : \ |C| \geq 45s \}$.

To show $\mathcal{A}' \subseteq \{ C \in \mathcal{C} \ : \ |C| \geq 45s \}$, suppose for contradiction that there is some C not in $\mathcal{A}$ with $|C| \geq 45s$. Lemma 5.37 implies there exists a cluster $A' \in \mathcal{A}$ that intersects C such that $|A'| > |C| \geq 45s$. We know $\mathcal{C}$ is a partition so $A' \notin \mathcal{C}$, so this contradicts the fact that we already proved $\mathcal{A}' \subseteq \{ C \in \mathcal{C} \ : \ |C| \geq 45s \}$.

This completes the proof of Lemma 5.31.

5.5.6 Runtime

The runtime is dominated by that needed to check the third property. Algorithm 5.3 considers $O(n^{6s})$ different values the sets T and U, so this takes time $O(n^{12s})$.

Chapter 6

Correlation clustering experiments

The results presented in this chapter are joint work with Micha Elsner. They previously appreared in Elsner and Schudy [2009].

6.1 Introduction

Practical work has adopted one of three strategies for solving correlation clustering problmes. For a few specific tasks, one can restrict the problem so that it is efficiently solvable [Malioutov and Barzilay, 2006]. In most cases, however, this is impossible. Integer linear programming (ILP) can be used to solve the general problem optimally, but only when the number of data points is small. Beyond a few hundred points, the only available solutions are heuristic or approximate.

In this paper, we evaluate a variety of solutions for correlation clustering on two realistic NLP tasks, text topic clustering and chat disentanglement, where typical datasets are too large for ILP to find a solution. We investigate the relationship between the clustering objective and external evaluation metrics such as F-score and one-to-one overlap, showing that optimizing the objective is usually a reasonable aim, but that other measurements like number of clusters found should sometimes be used to reject pathological solutions. We prove that the best heuristics are quite close to optimal, using the first implementation of the semi-definite programming (SDP) relaxation to provide tighter bounds.

6.2 Algorithms

We begin with some notation and a formal definition of the problem. Our input is a complete, undirected graph G with n nodes; each edge in the graph has a probability p_{ij} reflecting our belief as to whether nodes i and j come from the same cluster. Our goal is to find a clustering, defined as a new graph G' with edges $x_{ij} \in \{0, 1\}$, where if $x_{ij} = 1$, nodes i and j are assigned to the same

cluster. To make this consistent, the edges must define an equivalence relationship: $x_{ii} = 1$ and $x_{ij} = x_{jk} = 1$ implies $x_{ij} = x_{ik}$.

Our objective is to find a clustering as consistent as possible with our beliefs—edges with high probability should not cross cluster boundaries, and edges with low probability should. We define w_{ij}^+ as the cost of cutting an edge whose probability is p_{ij} and w_{ij}^- as the cost of keeping it. Mathematically, this objective can be written Ailon and Mohri [2008], Finkel and Manning [2008] as:

$$\min \sum_{ij:i<j} x_{ij} w_{ij}^- + (1 - x_{ij}) w_{ij}^+. \tag{6.1}$$

There are two plausible definitions for the costs w^+ and w^-, both of which have gained some support in the literature. We can take $w_{ij}^+ = p_{ij}$ and $w_{ij}^- = 1 - p_{ij}$ (*additive* weights) as in Ailon and Mohri [2008] and others, or $w_{ij}^+ = \log(p_{ij}), w_{ij}^- = \log(1-p_{ij})$ (*logarithmic* weights) as in Finkel and Manning [2008]. The logarithmic scheme has a tenuous mathematical justification, since it selects a maximum-likelihood clustering under the assumption that the p_{ij} are independent and identically distributed given the status of the edge ij in the true clustering. If we obtain the p_{ij} using a classifier, however, this assumption is obviously untrue—some nodes will be easy to link, while others will be hard—so we evaluate the different weighting schemes empirically.

6.2.1 Greedy Methods

We use four greedy methods drawn from the literature; they are all fast and easy to implement. All of them make decisions based on the *net weight* $w_{ij}^{\pm} = w_{ij}^+ - w_{ij}^-$.

These algorithms step through the nodes of the graph according to a permutation π. We try 100 random permutations for each algorithm and report the run which attains the best objective value (typically this is slightly better than the average run; we discuss this more in the experimental sections). To simplify the pseudocode we label the vertices $1, 2, \ldots n$ in the order specified by π. After this relabeling $\pi(i) = i$ so π need not appear explicitly in the algorithms.

Three of the algorithms are given in Figure 6.1. All three algorithms start with the empty clustering and add the vertices one by one. The BEST algorithm adds each vertex i to the cluster with the strongest $w^{\pm}$ connecting to i, or to a new singleton if none of the $w^{\pm}$ are positive. The FIRST algorithm adds each vertex i to the cluster containing the most recently considered vertex j with $w_{ij}^{\pm} > 0$. The VOTE algorithm adds each vertex to the cluster that minimizes the correlation clustering objective, i.e. to the cluster maximizing the total net weight or to a singleton if no total is positive.

6.2.2 Local Search

We use the straightforward local search previously used by Gionis et al. [2007] and Goder and Filkov [2008]. The allowed *one element moves* consist of removing one vertex from a cluster and either moving it to another cluster or to a new singleton cluster. The best one element move (BOEM) algorithm repeatedly makes the most profitable best one element move until a local optimum is

```
k ← 0 {number of clusters created so far}
for i = 1...n do
  for c = 1...k do
    if BEST then
      Quality_c ← max_{j∈C[c]} w_ij^±
    else if FIRST then
      Quality_c ← max_{j∈C[c]:w_ij^±>0} j
    else if VOTE then
      Quality_c ← Σ_{j∈C[c]} w_ij^±
    end if
  end for
  c* ← arg max_{1≤c≤k} Quality_c
  if Quality_{c*} > 0 then
    C[c*] ← C[c*] ∪ {i}
  else
    C[k++] ← {i} {form a new cluster}
  end if
end for
```

Figure 6.1: BEST/FIRST/VOTE algorithms

reached. *Simulated Annealing* (SA) makes a random single-element move, with probability related to the difference in objective it causes and the current temperature. Our annealing schedule is exponential and designed to attempt $2000n$ moves for n nodes. We initialize the local search either with all nodes clustered together, or at the clustering produced by one of our greedy algorithms (in our tables, the latter is written, eg. PIVOT/BOEM, if the greedy algorithm is PIVOT).

6.3 Bounding with SDP

Although comparing different algorithms to one another gives a good picture of relative performance, it is natural to wonder how well they do in an absolute sense—how they compare to the optimal solution. For very small instances, we can actually find the optimum using ILP, but since this does not scale beyond a few hundred points (see Section 6.4.1), for realistic instances we must instead bound the optimal value. Bounds are usually obtained by solving a *relaxation* of the original problem: a simpler problem with the same objective but fewer constraints.

The bound used in previous work Goder and Filkov [2008], Gionis et al. [2007], Bertolacci and Wirth [2007], which we call the *trivial bound*, is obtained by ignoring the transitivity constraints entirely. To optimize, we link ($x_{ij} = 1$) all the pairs where w_{ij}^+ is larger than w_{ij}^-; since this solution is quite far from being a clustering, the bound tends not to be very tight.

To get a better idea of how good a real clustering can be, we use a semi-definite programming (SDP) relaxation to provide a better bound. Here we motivate and define this relaxation.

6.4 Experiments

6.4.1 Scalability

Using synthetic data, we investigate the scalability of the linear programming solver and SDP bound. To find optimal solutions, we pass the complete ILP[1] to CPLEX. This is reasonable for 100 points and solvable for 200; beyond this point it cannot be solved due to memory exhaustion. As noted below, despite our inability to compute the LP bound on large instances, we can sometimes prove that they must be worse than SDP bounds, so we do not investigate LP-solving techniques further.

The SDP has fewer constraints than the ILP ($O(n^2)$ vs $O(n^3)$), but this is still more than many SDP solvers can handle. For our experiments we used one of the few SDP solvers that can handle such a large number of constraints: Christoph Helmberg's ConicBundle library Helmberg [2009, 2000]. This solver can handle several thousand datapoints. It produces loose lower-bounds (off by a few percent) quickly but converges to optimality quite slowly; we err on the side of inefficiency by running for up to 60 hours. Of course, the SDP solver is only necessary to bound algorithm performance; our solvers themselves scale much better.

6.4.2 Twenty Newsgroups

In this section, we test our approach on a typical benchmark clustering dataset, 20 Newsgroups, which contains posts from a variety of Usenet newsgroups such as `rec.motorcycles` and `alt.atheism`. Since our bounding technique does not scale to the full dataset, we restrict our attention to a subsample of 100 messages[2] from each newsgroup for a total of 2000—still a realistically large-scale problem. Our goal is to cluster messages by their newsgroup of origin. We conduct experiments by holding out four newsgroups as a training set, learning a pairwise classifier, and applying it to the remaining 16 newsgroups to form our affinity matrix.[3]

Our pairwise classifier uses three types of features previously found useful in document clustering. First, we bucket all words[4] by their log document frequency (for an overview of TF-IDF see Joachims [1997]). For a pair of messages, we create a feature for each bucket whose value is the proportion of shared words in that bucket. Secondly, we run LSA Deerwester et al. [1990] on the TF-IDF matrix for the dataset, and use the cosine distance between each message pair as a feature. Finally, we use the same type of shared words features for terms in message subjects. We make a training instance for each pair of documents in the training set and learn via logistic regression.

The classifier has an average F-score of 29% and an accuracy of 88%—not particularly good. We should emphasize that the clustering task for 20 newsgroups is much harder than the more common classification task—since our training set is entirely disjoint with the testing set, we can only learn weights on feature categories, not term weights. Our aim is to create realistic-looking data on which

[1] Consisting of the objective plus constraints $0 \leq x_{ij} \leq 1$ and triangle inequality Ailon and Mohri [2008].

[2] Available as `mini_newsgroups.tar.gz` from the UCI machine learning repository.

[3] The experiments below are averaged over four disjoint training sets.

[4] We omit the message header, except the subject line, and also discard word types with fewer than 3 occurrences.

Logarithmic Weights				
	Obj	Rand	F	1-1
SDP bound	51.1%	-	-	-
VOTE/BOEM	55.8%	93.80	33	41
SA	56.3%	93.56	31	36
PIVOT/BOEM	56.6%	93.63	32	39
BEST/BOEM	57.6%	93.57	31	38
FIRST/BOEM	57.9%	93.65	30	36
VOTE	59.0%	93.41	29	35
BOEM	60.1%	93.51	30	35
PIVOT	100%	90.85	17	27
BEST	138%	87.11	20	29
FIRST	619%	40.97	11	8

Table 6.1: Score of the solution with best objective for each solver, averaged over newsgroups training sets, sorted by objective.

to test our clustering methods, not to motivate correlation clustering as a solution to this specific problem. In fact, Zhong and Ghosh [2003] report better results using generative models.

We evaluate our clusterings using three different metrics (see Meila [2007] for an overview of clustering metrics). The *Rand* measure counts the number of pairs of points for which the proposed clustering agrees with ground truth. This is the metric which is mathematically closest to the objective. However, since most points are in different clusters, any solution with small clusters tends to get a high score. Therefore we also report the more sensitive *F-score* with respect to the minority ("same cluster") class. We also report the *one-to-one* score, which measures accuracy over single points. For this metric, we calculate a maximum-weight matching between proposed clusters and ground-truth clusters, then report the overlap between the two.

When presenting objective values, we locate them within the range between the trivial lower bound discussed in Section 6.3 and the objective value of the singletons clustering ($x_{ij} = 0, i \neq j$). On this scale, lower is better; 0% corresponds to the trivial bound and 100% corresponds to the singletons clustering. It is possible to find values greater than 100%, since some particularly bad clusterings have objectives worse than the singletons clustering. Plainly, however, real clusterings will not have values as low as 0%, since the trivial bound is so unrealistic.

Our results are shown in Table 6.1. The best results are obtained using logarithmic weights with VOTE followed by BOEM; reasonable results are also found using additive weights, and annealing, VOTE or PIVOT followed by BOEM. On its own, the best greedy scheme is VOTE, but all of them are substantially improved by BOEM. First-link is by far the worst. Our use of the SDP lower bound rather than the trivial lower-bound of 0% reduces the gap between the best clustering and the lower bound by over a factor of ten. It is easy to show that the LP relaxation can obtain a bound of at most 50%[5]—the SDP beats the LP in both runtime and quality!

[5]The solution $x_{ij} = \frac{1}{2}\mathbb{1}\left(w_{ij}^- > w_{ij}^+\right)$ for $i < j$ is feasible in the LP.

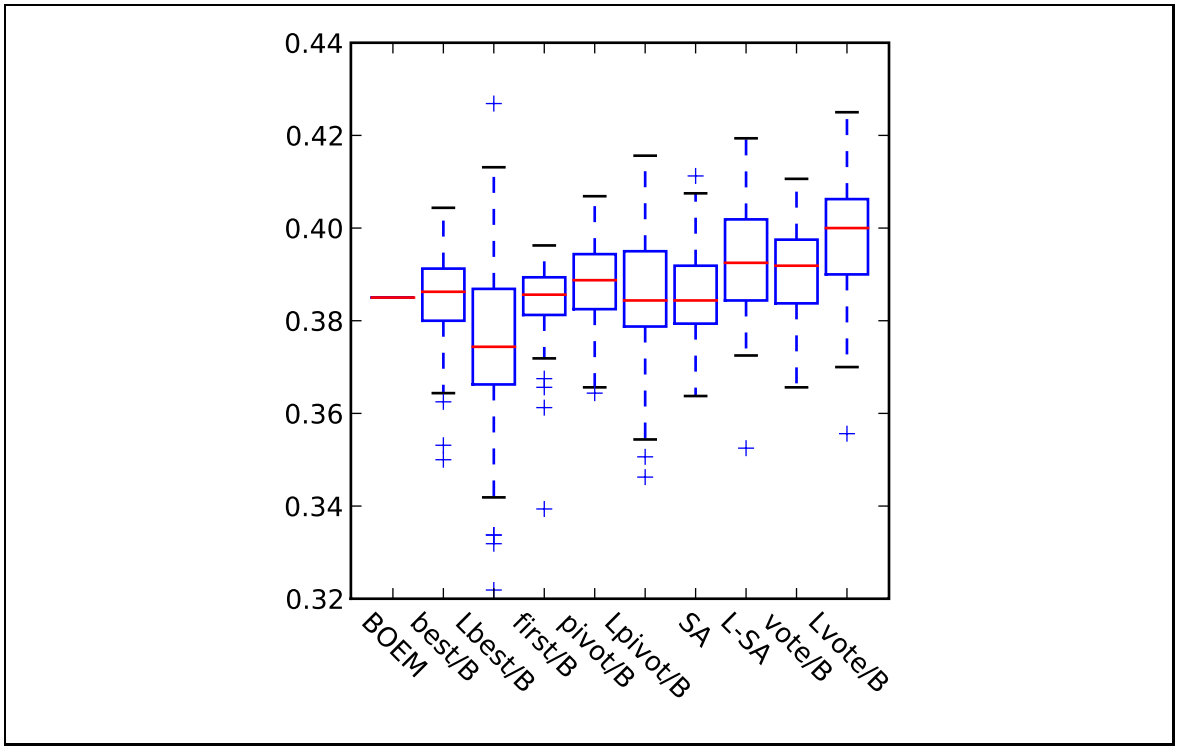

Figure 6.2: Box-and-whisker diagram (outliers as +) for one-to-one scores obtained by the best few solvers on a particular newsgroup dataset. L means using log weights. B means improved with BOEM.

We analyze the correlation between objective values and metric values, averaging Kendall's tau[6] over the four datasets (Table 6.2). Over the entire dataset, correlations are generally good (large and negative), showing that optimizing the objective is indeed a useful way to find good results. We also examine correlations for the solutions with objective values within the top 10%. Here the correlation is much poorer; selecting the solution with the best objective value will not necessarily optimize the metric, although the correspondence is slightly better for the log-weights scheme. The correlations do exist, however, and so the solution with the best objective value is typically slightly better than the median.

In Figure 6.2, we show the distribution of one-to-one scores obtained (for one specific dataset) by the best solvers. From this diagram, it is clear that log-weights and VOTE/BOEM usually obtain the best scores for this metric, since the median is higher than other solvers' upper quartile scores. All solvers have quite high variance, with a range of about 2% between quartiles and 4% overall. We omit the F-score plot, which is similar, for space reasons.

[6]The standard Pearson correlation coefficient is less robust to outliers, which causes problems for this data.

	Rand	F	1-1
Log-wt	-.60	-.73	-.71
Top 10 %	-.14	-.22	-.24
Add-wt	-.60	-.67	-.65
Top 10 %	-.13	-.15	-.14

Table 6.2: Kendall's tau correlation between objective and metric values, averaged over newsgroup datasets, for all solutions and top 10% of solutions.

6.4.3 Chat Disentanglement

In the disentanglement task, we examine data from a shared discussion group where many conversations are occurring simultaneously. The task is to partition the utterances into a set of conversations. This task differs from newsgroup clustering in that data points (utterances) have an inherent linear order. Ordering is typical in discourse tasks including topic segmentation and coreference resolution.

We use the annotated dataset and pairwise classifier made available by Elsner and Charniak [2008];[7] this study represents a competitive baseline, although more recently Wang and Oard [2009] have improved it. Since this classifier is ineffective at linking utterances more than 129 seconds apart, we treat all decisions for such utterances as abstentions, $p = .5$. For utterance pairs on which it does make a decision, the classifier has a reported accuracy of 75% with an F-score of 71%.

6.5 Conclusions

It is clear from these results that heuristic methods can provide good correlation clustering solutions on datasets far too large for ILP to scale. The particular solver chosen[8] has a substantial impact on the quality of results obtained, in terms of external metrics as well as objective value.

For general problems, our recommendation is to use log weights and run VOTE/BOEM. This algorithm is fast, achieves good objective values, and yields good metric scores on our datasets. Although objective values are usually only weakly correlated with metrics, our results suggest that slightly better scores can be obtained by running the algorithm many times and returning the solution with the best objective. This may be worth trying even when the datapoints are inherently ordered, as in chat.

Whatever algorithm is used to provide an initial solution, we advise the use of local search as a post-process. BOEM always improves both objective and metric values over its starting point.

The objective value is not always sufficient to select a good solution (as in the chat dataset). If possible, experimenters should check statistics like the number of clusters found to make sure they conform roughly to expectations. Algorithms that find far too many or too few clusters, regardless of objective, are unlikely to be useful. This type of problem can be especially dangerous if the pairwise

[7] Downloaded from `cs.brown.edu/~melsner`

[8] Our C++ correlation clustering software and SDP bounding package are available for download from `cs.brown.edu/~melsner`.

classifier abstains for many pairs of points.

SDP provides much tighter bounds than the trivial bound used in previous work, although how much tighter varies with dataset (about 12 times smaller for newsgroups, 3 times for chat). This bound can be used to evaluate the absolute performance of our solvers; the VOTE/BOEM solver whose use we recommend is within about 5% of optimality. Some of this 5% represents the difference between the bound and optimality; the rest is the difference between the optimum and the solution found. If the bound were exactly optimal, we could expect a significant improvement on our best results, but not a very large one—especially since correlation between objective and metric values grows weaker for the best solutions. While it might be useful to investigate more sophisticated local searches in an attempt to close the gap, we do not view this as a priority.

Chapter 7

Feedback arc set

The results presented in this chapter are joint work with Claire Mathieu. They previously appreared in conference form in Kenyon-Mathieu and Schudy [2007] and draft journal form in Mathieu and Schudy [2009].

7.1 Introduction

For general directed graphs, the FAS problem consists of removing the fewest number of edges so as to make the graph acyclic, and has applications such as scheduling [Flood, 1990] and graph layout [Sugiyama et al., 1981, Demetrescu and Finocchi, 2000] (see also [Lempel and Cederbaum, 1966, Baker and Hubert, 1977, Jünger, 1985]). This problem has been much studied, both in the mathematical programming community [Grötschel et al., 1985a,b, Jünger, 1985, Korte, 1979, Newman and Vempala, 2001, Newman, 2004], the approximation algorithms community [Leighton and Rao, 1999, Seymour, 1995, Even et al., 2000, 1998] and various applied communities [Demetrescu and Finocchi, 2000, 2001, Eades et al., 1993, Dwork et al., 2001]. The best known approximation ratio is $O(\log n \log\log n)$. Unfortunately the problem is NP-hard to approximate better than 1.36 [Karp, 1972, Dinur and Safra, 2002]. The equivalent problem of maximizing the number of edges not removed, called *max acyclic subgraph*, has also been extensively studied [Hassin and Rubinstein, 1994, Berger and Shor, 1997, Charikar et al., 2007a, Newman, 2001].

A *tournament* is the special case of directed graphs where every pair of vertices is connected by exactly one of the two possible directed edges. For tournaments, the FAS problem also has a long history, starting in the early 1960s in combinatorics [Seshu and Reed, 1961, Younger, 1963] and statistics [Slater, 1961]. In combinatorics and discrete probability, much early work [Erdös and Moon, 1965, Reid, 1969, Reid and Parker, 1970, Jung, 1970, Spencer, 1971, 1980, Fernandez de la Vega, 1983] focused on worst-case tournaments. In statistics and psychology, one motivation is *ranking by paired comparisons* [Slater, 1961]: here, you wish to sort some set by some objective but you do not

have access to the objective, only a way to compare a pair and see which is greater; for example, determining people's preferences for types of food. Feedback arc set in tournament graphs and closely related problems have also been used in machine learning [Cohen et al., 1999, Ailon and Mohri, 2008]. Unfortunately the FAS problem is NP-hard even in tournaments [Ailon et al., 2008, Alon, 2006, Charbit et al., 2007] (see also [Conitzer, 2006]).

Researchers have been designing algorithms for the FAS problem in tournament graphs since the dawn of computer science. Slater and Alway describe simple heuristics for the FAS tournament problem in Slater [1961] (for comparison Hoare published quicksort the same year). Coleman and Wirth [2008] compare various algorithms empirically, including an algorithm by Ailon et al. [2008] and many others with no approximation guarantees. The best previously known approximation algorithms achieve constant factor approximations: 2.5 in the randomized setting [Ailon et al., 2008] and 3 in the deterministic setting [Van Zuylen et al. 2007; Van Zuylen and Williamson 2007; Ailon et al. 2008] (see also Coppersmith et al. [2006]). Our main result is a polynomial time approximation scheme (PTAS) for this problem: thus the problem really is provably easier to approximate in tournaments than on general graphs.

Here is a weighted generalization of feedback arc set in tournaments.

Problem 7.1 (Weighted FAS Tournament)**.**
Input: complete directed graph with vertex set V*,* $n \equiv |V|$ *and non-negative edge weights* w_{uv} *with* $b \leq w_{uv} + w_{vu} \leq 1$ *for some fixed positive constant* b*.*
Output: An ranking π *minimizing* $C(\pi) = \sum_{\{u,v\} \subset V : \pi(v) > \pi(u)} w_{vu}$*, where a* ranking *is a bijective mapping from* V *to* $\{1, 2, \ldots |V|\}$*.*

(The unweighted problem is the special case where $w_{uv} = 1$ if (u, v) is an edge and $w_{uv} = 0$ otherwise.)

Theorem 7.2 (PTAS)**.** *There is a randomized algorithm for minimum Feedback Arc Set on weighted tournaments. Given* $\epsilon > 0$*, it outputs a ranking with expected cost at most* $(1+\epsilon)OPT$*. The expected running time is:*

$$O\left(n^3 \log n(\log(1/b) + 1/\epsilon)\right) + n2^{\tilde{O}(1/(\epsilon b)^6)}.$$

The algorithm can be derandomized at the cost of increasing the running time to $n^{\tilde{O}(1/(\epsilon b)^{12})}$*.*

We remark that our PTAS is singly exponential in $1/\epsilon$, whereas the PTAS in the conference version of this work [Kenyon-Mathieu and Schudy, 2007] was doubly exponential in $1/\epsilon$.

Ailon et al. [2008] study the special-case $b = 1$, which they call *weighted FAS tournament with probability constraints*. Indeed, sampling a population naturally leads to defining w_{ij} as the probability that type i is preferred to type j. We note that all known approximation algorithms [Ailon et al. 2008; Coppersmith et al. 2006; Van Zuylen et al. 2007; Van Zuylen and Williamson 2007] extend to weighted tournaments for $b = 1$.

An important application of weighted FAS tournaments is *rank aggregation*. Frequently, one has access to several rankings of objects of some sort, such as search engine outputs [Dwork et al., 2001],

and desires to aggregate the input rankings into a single output ranking that is similar to all of the input rankings: it should have minimum average distance from the input rankings, for some notion of distance. This ancient problem was already studied in the context of voting by Borda [1781] and Condorcet [1785] in the 18^{th} century, and has aroused renewed interest recently [Dwork et al., 2001, Conitzer et al., 2006]. A natural notion of distance is the number of pairs of vertices that are in different orders: this defines the *Kemeny rank aggregation* problem [Kemeny, 1959, Kemeny and Snell, 1962]. This choice yields a maximum likelihood estimator for a certain naïve Bayes model [Young, 1995]. This problem is NP-hard [Bartholdi et al., 1989], even with only 4 voters [Dwork et al., 2001]. Constant factor approximation algorithms are known: choosing a random (or best) input ranking as the output gives a 2-approximation; finding the optimal aggregate ranking using the footrule distance as the metric instead of the Kendall-Tau distance also gives a 2-approximation [Dwork et al., 2001, Diaconis and Graham, 1977]. There is also a randomized 4/3 approximation algorithm [Ailon et al., 2008]. We improve on these results by giving a polynomial time approximation scheme (PTAS).

Corollary 7.3. *There is a randomized algorithm for Kemeny rank aggregation that. Given $\epsilon > 0$, it outputs a ranking with expected cost at most $(1+\epsilon)OPT$. The expected running time for n candidates is:*

$$O\left(\frac{n^3 \log n}{\epsilon}\right) + n2^{\tilde{O}(1/\epsilon^6)} + O\left(n^2 \cdot (\textit{number of voters})\right).$$

The algorithm can be derandomized at the cost of increasing the running time to $n^{\tilde{O}(1/\epsilon^{12})}$.

An important open question is whether there is a PTAS for the generalization of Kemeny rank aggregation to *partial* rankings such as search engine outputs. There is a 1.5-approximation algorithm [Ailon, 2007].

It is surprising that the minimum Feedback Arc Set problem on tournaments has an approximation scheme. The related problems of correlation clustering on complete graphs[1] [Charikar et al., 2005] and of feedback *vertex* set on tournaments [Cai et al., 2001] do not have PTASs unless P=NP.

Certain special cases can be solved exactly in polynomial time. Braverman and Mossel [2008] give an exact algorithm for feedback arc set when the input is generated by adding noise to a base ranking. There are also good algorithms for low-cost instances of FAS tournament and Kemeny rank aggregation (i.e. fixed-parameter tractability) [Alon et al., 2009, Dom et al., 2006, Betzler et al., 2008]. We will show an improvement in the runtime of FAST tournament from $2^{\tilde{O}(\sqrt{OPT})}$ to $2^{O(\sqrt{OPT})}$. We will investigate fixed parameter tractability of Kemeny rank aggregation and fully dense betweenness and expect to improve those runtimes as well.

Other related problems include d-dimensional arrangement [Charikar et al., 2007b].

We note that Amit Agarwal has informed us that he has obtained similar results.

[1] This problem is identical to FAS except it deals with symmetric transitive relations instead of antisymmetric ones.

KwikSort(vertices S):

Choose a vertex v uniformly at random from S
Let L be the set of vertices u such that $w_{uv} \geq w_{vu}$ and $R = S \setminus L$.
Return the concatenation of KwikSort(L) and KwikSort(R).

Figure 7.1: KwikSort algorithm for minimum Feedback Arc Set in tournaments by Ailon et al. [2008].

7.2 Main Results

7.2.1 Algorithmic tools

Our first algorithmic tool is local search. A *single vertex move*, given an ranking π, a vertex x and a position i, consists of taking x out of π and putting it back in position i. We say a ranking is *locally optimal* if no single vertex move can improve its cost. Single vertex moves were used for FAS tournament as early as 1961 [Slater, 1961]. Single vertex moves and variants were also used in [Hassin and Rubinstein, 1994, Younger, 1963, Dwork et al., 2001, Coleman and Wirth, 2008]. Coleman and Wirth [2008] show that single vertex moves alone do not yield a constant factor approximation by giving a graph with global optimum $\Theta(n)$ and a local optimum of value $\Theta(n^2)$.

Our second algorithmic tool is the KwikSort algorithm by Ailon Charikar and Newman Ailon et al. [2008]. Recall from Problem 7.1 that b is a lower bound on $w_{uv} + w_{vu}$ for every pair $\{u, v\}$. We show that Ailon Charikar and Newman's [2008] KwikSort algorithm, (which we reproduce for completeness in Figure 7.1) is a $5/b$-approximation for any $b > 0$.

Theorem 7.4. *[Ailon et al., 2008] Let w be non-negative weight function on the edges. Assume that for every $u, v, x \in V$, if $w_{uv} \geq w_{vu}$, $w_{vx} \geq w_{xv}$ and $w_{xu} \geq w_{ux}$, then $w_{uv} + w_{vx} + w_{xu} \leq \alpha \cdot \min\{w_{uv}, w_{vx}, w_{xu}\}$ for some $\alpha > 1$. Then the* KwikSort *algorithm is an α-approximation in expectation.*

Corollary 7.5. *Assume that for every pair of vertices, $b \leq w_{uv} + w_{vu} \leq 1$. Then the* KwikSort *algorithm is a $5/b$-approximation in expectation.*

Proof. (Adapted from the proof in Ailon et al. [2008] for the case $b = 1$.) Fix some triple $\{u, v, x\} \subseteq V$ with $w_{uv} \geq w_{vu}$, $w_{vx} \geq w_{xv}$ and $w_{xu} \geq w_{ux}$. We assume without loss of generality that $w_{uv} \leq w_{vx}, w_{xu}$. By weighted tournament assumption $w_{uv} + w_{vu} \geq b$ so $w_{uv} \geq b/2$, hence $2 \leq \frac{4}{b} w_{uv}$.

Therefore

$$\begin{aligned} w_{uv} + w_{vx} + w_{xu} &\leq 2 + w_{uv} \leq \left(\frac{4}{b} + 1\right) w_{uv} = \left(\frac{4}{b} + 1\right) \min\{w_{uv}, w_{vx}, w_{xu}\} \\ &\leq \frac{5}{b} \min\{w_{uv}, w_{vx}, w_{xu}\}. \end{aligned}$$

□

Our third and last algorithmic tool is the sampling-based approximation algorithms due to Arora,

FAST-SCHEME:

Run the KWIKSORT algorithm on V to define an ranking π
Return IMPROVE(π, $\epsilon b/5$).

IMPROVE(ranking π, error tolerance η):

Set $\beta \leftarrow \frac{\eta C(\pi)}{4n \log_{3/2} n}$ and $\kappa \leftarrow \frac{\eta^2 b^3}{350 \cdot 400^2}$
Perform single vertex moves on π until none can improve the cost by more than β.
Return IMPROVEREC(V, π)

IMPROVEREC(vertices S, ranking π on S):
if $|S| = 1$ **then**
Return π
else
if $C(\pi) \geq \kappa |S|^2$ **then**
Return the ranking from ADDAPPROX with $\delta = \frac{\eta}{4}\kappa$
else
Choose an integer k uniformly at random from $[|S|/3, 2|S|/3]$
Let L be the set of vertices v such that $\pi(v) \leq k$ and $R = S \setminus L$
Return concatenation of IMPROVEREC(L, π_L) and IMPROVEREC(R, π_R) where
π_L is the ranking of L induced by π, i.e. $\pi_L(v) = \pi(v)$, and
π_R is the ranking of R induced by π, i.e. $\pi_R(v) = \pi(v) - k$.
end if
end if

Figure 7.2: Approximation scheme for minimum Feedback Arc Set on tournaments

Frieze and Kaplan 2002 and to Frieze and Kannan 1999 described in Section 2. For completeness we use a (much simpler) algorithm based on our work reported in Section 2.

Theorem 7.6. *Let $\delta > 0$ be fixed. There is a randomized algorithm* ADDAPPROX *for minimum Feedback Arc Set on weighted tournaments with expected additive error δn^2 and runtime $O\left(n^2\right) + 2^{\tilde{O}(1/\delta^2)}$.*

7.2.2 Algorithm

Our main algorithm FAST-SCHEME (short for Feedback Arc Set Tournament Approximation Scheme) is presented in Figure 7.2.

Lemma 7.7. *Let π^{in} be the input and π^{out} be the output of the* IMPROVE *subroutine. Then:*

$$\mathbf{E}\left[C(\pi^{out})\right] \leq OPT + \eta \cdot C(\pi^{in}).$$

With this, it is easy to see that FAST-SCHEME is an approximation scheme. Indeed, by Corollary 7.5, it calls the IMPROVE subroutine for a ranking of expected cost at most $(5/b)OPT$. By definition of η and Lemma 7.7, the output ranking then has expected cost at most $(1+\epsilon)OPT$. To obtain the running time claimed in Theorem 7.2, we actually need a slightly modified version of FAST-SCHEME that calls IMPROVE repeatedly (see Section 7.5.)

Definition 7.8. *Ranking π of S* respects *partition L, R of S if $\pi(u) < \pi(v)$ for every $u \in L$ and $v \in R$.*

The keystone of our analysis is the following Lemma.

Lemma 7.9. *Let $\beta \geq 0$ and π^{loc} be an ranking such that no single vertex move can improve the cost by more than β. Let k be an integer chosen uniformly at random from $[|S|/3, 2|S|/3]$, L be the set of vertices such that $\pi^{loc}(v) \leq k$ and $R = S \setminus L$. Let π^* be an optimum ranking and π' be an optimum ranking that respects L, R. We have*

$$\mathbf{E}\left[C(\pi')\right] \leq C(\pi^*) + \frac{400}{|S|}\left(\frac{C(\pi^{loc})}{b}\right)^{3/2} + \beta|S|$$

7.2.3 Why FAST-Scheme needs single vertex moves

As a warm-up to build intuition we demonstrate that FAST-Scheme needs single vertex moves. Consider a variant of FAST-Scheme that calls ImproveRec directly on the output of KwikSort, without doing any single vertex moves. We now show that this variant is not a PTAS.

Consider a chess tournament (instance) where all the results are consistent except that the worst player beat the best. If KwikSort picks any player other than the worst or best as the first pivot it finds a ranking with cost 1, which is optimal. On the other hand, if KwikSort picks the best player as the first pivot it produces ranking π^{bad}, which puts the worst player first and has cost $n - 1$.

Now consider what happens when ImproveRec is called on π^{bad}. As long as n is not too small ImproveRec divides and conquers, irrevocably committing to ranking the worst player among the k best for some $n/3 \leq k \leq 2n/3$. No matter what happens in the later recursions, the output clearly has cost at least $n/3$.

The overall expected cost of the output of this variant of FAST-Scheme is therefore at least

$$\frac{n-2}{n}1 + \frac{2}{n}\frac{n}{3} = \frac{5}{3} + o(1)$$

hence this variant is not a PTAS.

7.2.4 Organization of this chapter

The core of this chapter is §7.3, which proves Lemma 7.9. In §7.4 we use Lemma 7.9 to prove Lemma 7.7, completing our proof that FAST-Scheme is an approximation scheme. In §7.5 we describe a variant of FAST-Scheme with better runtime and prove Theorem 7.2. In §7.6 we derandomize. An appendix describes the AddApprox algorithm promised by Theorem 7.6.

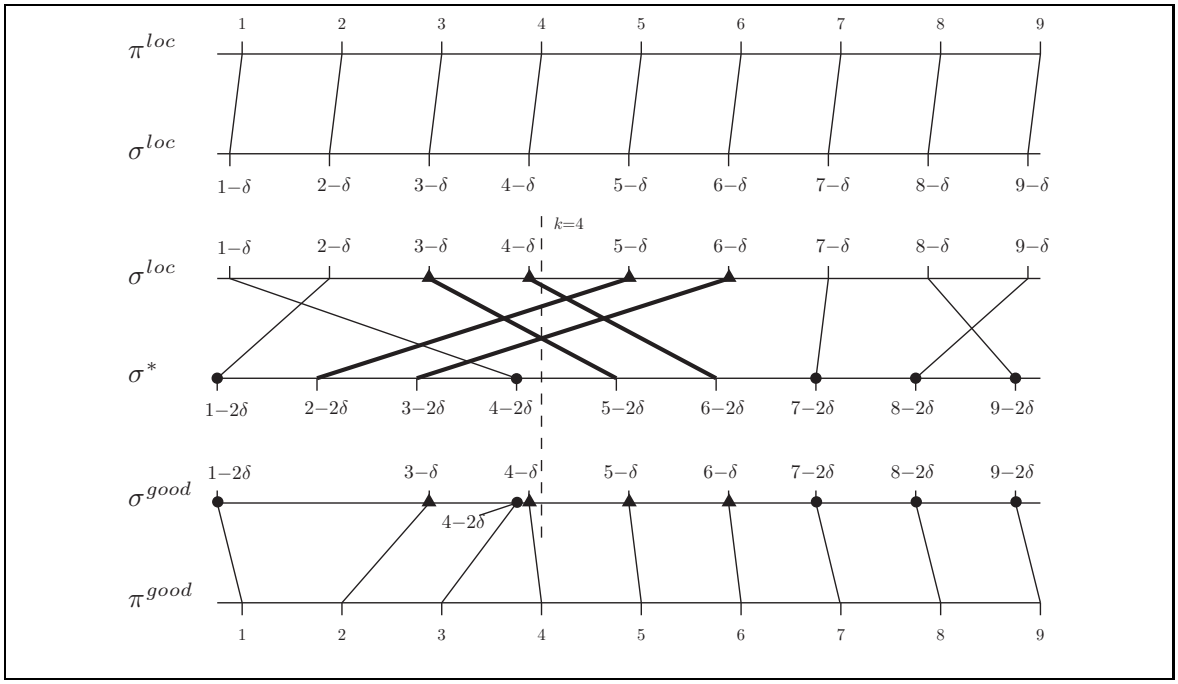

Figure 7.3: Definition of π^{good}: edges such that k is between $\sigma^{loc}(u)$ and $\sigma^*(u)$ are in bold.

7.3 Bounding the cost of partition-respecting rankings

7.3.1 A good partition-respecting ranking

In preparation for proving Lemma 7.9 we adopt its notation. We modify π^* to construct a partition-respecting ranking π^{good} as follows. To avoid having to introduce cumbersome tie-breaking rules, we shift values a little: let $\delta = 1/100$, $\sigma^{loc}(v) = \pi^{loc}(v) - \delta$ and $\sigma^*(v) = \pi^*(v) - 2\delta$. Let

$$\sigma^{good}(v) = \begin{cases} \sigma^*(v) & \text{if } \sigma^{loc}(v) \text{ and } \sigma^*(v) \text{ are on the same side of } k \\ \sigma^{loc}(v) & \text{otherwise} \end{cases}.$$

We then define π^{good} as the ranking naturally associated with σ^{good}: $\pi^{good}(v)$ is the rank of $\sigma^{good}(v)$ among $\{\sigma^{good}(u) : u \in S\}$. See Figure 7.3 for an illustration.

Lemma 7.10. *π^{good} respects (L, R).*

Proof. By definition of σ^{good}, $\sigma^{good}(u) < k$ if and only if $\sigma^{loc}(u) < k$, which holds if and only if $\pi^{loc}(u) \leq k$, or in other words, if and only if $u \in L$. □

7.3.2 Proof of Lemma 7.9

We call an injective map σ from S to $\mathbb{R}$ an *ordering*. Given ordering σ, we define Ranking(σ) as the ranking naturally associated to σ: Ranking(σ)(v) is the rank of $\sigma(v)$ among $\{\sigma(u) : u \in S\}$. For instance, Ranking(σ^{loc}) $= \pi^{loc}$, Ranking(σ^*) $= \pi^*$, and Ranking(σ^{good}) $= \pi^{good}$. The notion of

single vertex move extends naturally to orderings and consists of changing the value of the ordering at a single vertex. We also extend the notion of cost in the obvious way: $C(\sigma) = C(\text{Ranking}(\sigma))$.

For any vertex u, define σ_u^{loc} to be the result of applying a certain single vertex move to σ^{loc}, defined by $\sigma_u^{loc}(v) = \sigma^{loc}(v)$ for all v except for u and $\sigma_u^{loc}(u) = \sigma^*(u)$. For any $x \in \mathbb{R}$ we write u *crosses* x if x is between $\sigma^{loc}(u)$ and $\sigma^*(u)$.

Lemma 7.11. *Let* $T = \sum_{u\,:\,u \text{ crosses } k} \left(C(\sigma^{loc}) - C(\sigma_u^{loc})\right)$. *Then* $T \le \beta|S|$.

Proof. By definition of π^{loc} (the ranking associated with σ^{loc}), no single vertex move can improve the cost by more than β, so $C(\sigma^{loc}) - C(\sigma_u^{loc}) \le \beta$. Summing over u concludes the proof. □

Lemma 7.12. *Let* $F = \sum_{v \in S} |\pi^{loc}(v) - \pi^*(v)|$ *denote the* Spearman's footrule distance *between rankings* π^{loc} *and* π^*. *Then*

$$\mathbf{E}\left[C(\sigma^{good}) - C(\sigma^*) - T\right] \le \frac{32\sqrt{2}}{|S|} F^{3/2}$$

Before proving Lemma 7.12, let us see how it implies Lemma 7.9.

Proof. (of Lemma 7.9). By Lemma 7.10 and the definition of π' we have $C(\pi') \le C(\pi^{good})$, which equals $C(\sigma^{good})$ by the correspondence between rankings and orderings. By Lemma 7.12, in expectation this is at most

$$\mathbf{E}\left[C(\sigma^*) + T\right] + \frac{32\sqrt{2}}{|S|} F^{3/2}.$$

which we can in turn bound using Lemma 7.11 by

$$C(\sigma^*) + \beta|S| + \frac{32\sqrt{2}}{|S|} F^{3/2}$$

By Diaconis and Graham [1977]'s Theorem 2 relating the Spearman's footrule and Kendall-Tau metrics on rankings, we have:

$$\begin{aligned}
F &= \sum_v |\pi^{loc}(v) - \pi^*(v)| \\
&\le 2 \sum_{u,v} \mathbb{1}\left(\pi^*(u) > \pi^*(v) \text{ and } \pi^{loc}(u) < \pi^{loc}(v)\right) && \text{[D\&G 1977, Thm 2]} \\
&\le 2 \sum_{u,v} \left[\mathbb{1}\left(\pi^*(u) > \pi^*(v)\right) \frac{w_{uv}}{b} + \mathbb{1}\left(\pi^{loc}(u) < \pi^{loc}(v)\right) \frac{w_{vu}}{b}\right] && \text{since } w_{uv} + w_{vu} \ge b \\
&= (2/b)(C(\pi^*) + C(\pi^{loc})). && \text{by definition of } C \\
&\le (4/b) C(\pi^{loc}),
\end{aligned}$$

where $\mathbb{1}(p)$ is the indicator function of event p. Therefore

$$\mathbf{E}\left[C(\pi')\right] \quad \le \quad C(\sigma^*) + \beta|S| + \frac{32\sqrt{2}}{|S|} \left(\frac{4C(\pi^{loc})}{b}\right)^{3/2}.$$

By the correspondence between orderings and rankings again, $C(\sigma^*) = C(\pi^*)$, and clearly $32 \cdot \sqrt{2} \cdot 4^{3/2} < 400$, hence the Lemma. □

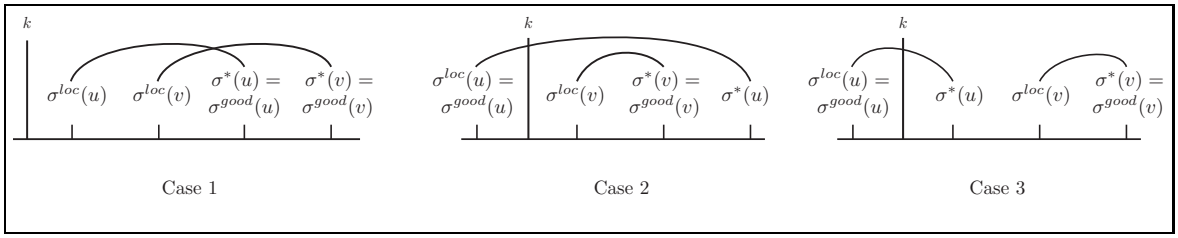

Figure 7.4: Illustration of cases in proof of Lemma 7.13

7.3.3 Proof of Lemma 7.12

For any $x, z \in \mathbb{R}$, define $(x, z) = \{\, y \in \mathbb{R} \;:\; x < y < z \text{ or } z < y < x \,\}$. We say $\{u, v\}$ is a *crossing pair* if the intervals $\big(\sigma^*(u), \sigma^{loc}(u)\big)$ and $\big(\sigma^*(v), \sigma^{loc}(v)\big)$ intersect but neither is contained within the other. We say that $\{u, v\}$ is a *cut crossing pair* if it is a crossing pair and $k \in \big(\sigma^*(u), \sigma^{loc}(u)\big) \cup \big(\sigma^*(v), \sigma^{loc}(v)\big)$.

Lemma 7.13.

$$C(\sigma^{good}) - C(\sigma^*) - T \leq 4|\{\{u, v\} \textit{ cut crossing pair}\}|$$

Proof. We use the following notation: given an ordering σ, let $C(\sigma)_{uv}$ denote the contribution of edge $\{u, v\}$ to the cost of σ, that is, w_{uv} or w_{vu} depending on the relative order of u and v. Let $t_{uv} = \mathbb{1}\,(u \text{ crosses } k)\left(C(\sigma^{loc})_{uv} - C(\sigma^{loc}_u)_{uv}\right)$. With these notations, it trivially follows that

$$C(\sigma^{good}) - C(\sigma^*) - T = \sum_{\{u,v\}} \left[C(\sigma^{good})_{uv} - C(\sigma^*)_{uv} - (t_{uv} + t_{vu})\right].$$

If $\{u, v\}$ is a cut-crossing pair, we use naïve bounds: $C(\sigma^{good})_{uv}, C(\sigma^*)_{uv}, t_{uv}$ and t_{vu} are all at most 1 in absolute value, and so

$$C(\sigma^{good})_{uv} - C(\sigma^*)_{uv} - (t_{uv} + t_{vu}) \leq 4.$$

If $\{u, v\}$ is not a cut crossing pair, we do a case-by-case analysis to prove that

$$C(\sigma^{good})_{uv} - C(\sigma^*)_{uv} - (t_{uv} + t_{vu}) = 0.$$

The three cases are illustrated in Figure 7.4.

Case 1: If neither u nor v cross k, then $C(\sigma^{good})_{uv} = C(\sigma^*)_{uv}$ by definition of σ^{good}, and $t_{uv} = t_{vu} = 0$ by definition of t.

Case 2: Suppose either v or u (or both) cross k and the intervals $\big(\sigma^*(u), \sigma^{loc}(u)\big)$ and $\big(\sigma^*(v), \sigma^{loc}(v)\big)$ are nested. Without loss of generality suppose $\big(\sigma^*(v), \sigma^{loc}(v)\big)$ is contained within $\big(\sigma^*(u), \sigma^{loc}(u)\big)$. It is impossible for v to cross k without u crossing k as well so we conclude u crosses k. Whether the image of v is $\sigma^{loc}(v)$ or $\sigma^*(v)$ does not affect the orientation of edge uv, and so $C(\sigma^{good})_{uv} = C(\sigma^{loc})_{uv}$, $C(\sigma^*)_{uv} = C(\sigma^{loc}_u)_{uv}$ and $C(\sigma^{loc}_v)_{uv} = C(\sigma^{loc})_{uv}$. Therefore by definitions $C(\sigma^{good})_{uv} - C(\sigma^*)_{uv} = t_{uv}$ and $t_{vu} = 0$.

Case 3: Suppose either v or u (or both) cross k and the intervals $\left(\sigma^*(u), \sigma^{loc}(u)\right)$ and $\left(\sigma^*(v), \sigma^{loc}(v)\right)$ are disjoint. The edge uv is oriented the same way in all of the relevant orderings so $C(\sigma^{good})_{uv} = C(\sigma^{loc})_{uv} = C(\sigma^*)_{uv} = C(\sigma^{loc}_u)_{uv} = C(\sigma^{loc}_v)_{uv}$. Therefore $C(\sigma^{good})_{uv} - C(\sigma^*)_{uv} = t_{uv} = t_{vu} = 0$.

This proves the Lemma. □

Lemma 7.14. *For any $u \in S$, the probability that u crosses k is at most*

$$\frac{4|\pi^{loc}(u) - \pi^*(u)|}{|S|}.$$

Proof. Let $K = \{\, k \;:\; |S|/3 \le k \le 2|S|/3 \,\}$ be the set k is chosen randomly from. The probability that u crosses k equals $|\{\, k \in K \;:\; u \text{ crosses } k \,\}|/|K|$. The numerator $|\{\, k \in K \;:\; u \text{ crosses } k \,\}|$ is at most $|\pi^{loc}(u) - \pi^*(u)|$. The denominator $|K|$ is approximately $|S|/3$, and a careful analysis for $|S|$ small shows that for $|S| > 1$ it is always at least $|S|/4$, hence the result. □

Lemma 7.15. *For any vertex $u \in S$, $|\{\, v \in S \;:\; \{u, v\} \text{ crossing pair} \,\}| \le 2\sqrt{2F}$*

Proof. Fix a vertex u. Observe that if $\{u, v\}$ is a crossing pair, then v must cross $\lceil \sigma^{loc}(u) \rceil$ or $\lfloor \sigma^*(u) \rfloor$ (the floor and ceiling come from the shift by $-\delta$ for σ^{loc} and by -2δ for σ^*). Thus

$$|\{\, v \in S \;:\; \{u, v\} \text{ crossing pair} \,\}| \;\;\le\;\; \xi(\lceil \sigma^{loc}(u) \rceil) + \xi(\lfloor \sigma^*(u) \rfloor), \tag{7.1}$$

where $\xi(j)$ denotes the number of vertices that cross integer j.

Now we need to relate $\xi(i)$ to the footrule distance F. We write:

$$F \;\;=\;\; \sum_u |\pi^{loc}(u) - \pi^*(u)| = \sum_u \sum_{j \in \mathbb{Z}} \mathbb{1}\left(j \in \left(\sigma^*(u), \sigma^{loc}(u)\right)\right) = \sum_{j \in \mathbb{Z}} \xi(j).$$

Now, it is easy to see that we always have $|\xi(j) - \xi(j+1)| \le 2$ (since the difference can only come from two vertices: the vertex such that $\sigma^{loc}(u) = j+1-\delta$, and the vertex such that $\sigma^*(u) = j+1-\delta$.) Thus for any i we have:

$$\sum_{j \in \mathbb{Z}} \xi(j) \ge \xi(i) + 2 \sum_{j=1}^{\lfloor \xi(i)/2 \rfloor} (\xi(i) - 2j) \ge \frac{\xi(i)^2}{2},$$

where the last inequality follows after a straightforward calculation. Thus for all i

$$\xi(i) \le \sqrt{2F} \tag{7.2}$$

Applying Equation (7.2) to $i = \lceil \sigma^{loc}(u) \rceil$ and to $i = \lfloor \sigma^*(u) \rfloor$, summing the results and plugging into Equation (7.1) yields the Lemma. □

Now we prove Lemma 7.12.

Proof. By Lemmas 7.13, 7.14 and 7.15

$$\begin{aligned}
&\mathbf{E}\left[C(\sigma^{good}) - C(\sigma^*) - T\right] \\
&\quad\le\;\; 4 \sum_u \Pr\left(u \text{ crosses } k\right) \cdot |\{\, v \in S \;:\; \{u, v\} \text{ crossing pair} \,\}| \\
&\quad\le\;\; 4 \sum_u \frac{4|\pi^{loc}(u) - \pi^*(u)|}{|S|} 2\sqrt{2F} = 32\sqrt{2}\frac{F^{3/2}}{|S|}.
\end{aligned}$$

□

7.4 Summing errors over the recursion tree (proof of Lemma 7.7)

Consider each set S on which we execute algorithm ImproveRec, and call such a set *leaf* or *internal* depending on whether or not there are further recursive calls to ImproveRec. Let π_S^{loc} and π_S^{out} denote respectively the input and output orderings of ImproveRec when run on S. Let π' denote the optimal ranking of S that respects (L, R), π'_L denote the optimal ranking of L (which is also the restriction of π' to L) and π'_R denote the optimal ranking of R (which is also the restriction of π' to R). Let π_L^{out} and π_R^{out} denote the restrictions of π^{out} to L and R respectively. The key observation is:

$$\begin{aligned}
&\mathbf{E}\left[C(\pi_S^{out})\right] - OPT_S \\
&\quad = \mathbf{E}\left[C(\pi_S^{out}) - C(\pi')\right] + \mathbf{E}\left[C(\pi') - OPT_S\right] \\
&\quad = \mathbf{E}\left[C(\pi_L^{out}) - C(\pi'_L)\right] + \mathbf{E}\left[C(\pi_R^{out}) - C(\pi'_R)\right] + \mathbf{E}\left[C(\pi') - OPT_S\right] \qquad (7.3)
\end{aligned}$$

because π^{out} and π' both respect L, R. To be more precise about the meanings of the expectations in(7.3) let $\mathbb{T}_S$ be a random variable expressing the random choices made by ImproveRec(S, π_S^{loc}) and all of its descendants. We restate (7.3) formally as:

$$\begin{aligned}
&\mathbf{E}_{\mathbb{T}_S}\left[C(\pi_S^{out})\right] - OPT_S \\
&\quad = \mathbf{E}_k\left[\mathbf{E}_{\mathbb{T}_L}\left[C(\pi_L^{out}) - OPT_L\right]\right] + \mathbf{E}_k\left[\mathbf{E}_{\mathbb{T}_R}\left[C(\pi_R^{out}) - OPT_R\right]\right] + \\
&\qquad + \mathbf{E}_k\left[C(\pi') - OPT_S\right] \qquad (7.4)
\end{aligned}$$

Now use Lemma 7.9 to write

$$\mathbf{E}_k\left[C(\pi') - OPT_S\right] \le \frac{400}{|S|}\left(\frac{C(\pi^{loc})}{b}\right)^{3/2} + \beta|S|. \qquad (7.5)$$

Applying (7.4) and (7.5) repeatedly over the recursion tree we get

$$\begin{aligned}
&\mathbf{E}_{\mathbb{T}_V}\left[C(\pi_V^{out})\right] - OPT \qquad (7.6)\\
&\quad \le \mathbf{E}_{\mathbb{T}_V}\left[\sum_{S \text{ leaf}} \mathbf{E}_{\mathbb{T}_S}\left[C(\pi_S^{out}) - OPT_S\right] + \sum_{S \text{ internal}} \beta|S| + \sum_{S \text{ internal}} \frac{400}{|S|}\left(\frac{C(\pi_S^{loc})}{b}\right)^{3/2}\right].
\end{aligned}$$

To complete the proof we bound the argument of the expectation that is the right hand side of (7.6) by $\eta C(\pi_V^{loc})$ uniformly over any set of random choices $\mathbb{T}_V$. Dealing with the first sum is straightforward: any leaf must have $\mathbf{E}\left[C(\pi_S^{out})\right] \le OPT_S + \frac{\eta}{4}\kappa|S|^2 \le OPT_S + \frac{\eta}{4}C(\pi_S^{loc})$. Summing,

$$\sum_{S \text{ leaf}} \mathbf{E}\left[C(\pi_S^{out}) - OPT_S\right] \le \frac{\eta}{4}\sum_{S \text{ leaf}} C(\pi_S^{loc}) \le \frac{\eta}{4}C(\pi_V^{loc}) \le \frac{\eta}{4}C(\pi_V^{in})$$

where π^{in} is the ranking input to Improve.

Dealing with the second sum in (7.6) is also straightforward: note that the tree of recursive calls has at most $\log_{3/2} n$ levels of internal nodes and that on each level the sets S are disjoint so

$$\sum_{S \text{ internal}} \beta|S| \leq \sum_{\text{levels}} \beta n \leq \beta n \log_{3/2} n \leq \frac{\eta}{8} C(\pi_V^{in}).$$

To deal with the third sum in (7.6), we start with two preliminary lemmas. Consider an internal node S and its two children L and R.

Lemma 7.16. *Let c_S, c_L, c_R denote $C(\pi_S^{loc})$, $C(\pi_L^{loc})$ and $C(\pi_R^{loc})$ respectively. Then:*

$$\frac{(c_S - c_L)^{3/2}}{|S| - |L|} - \frac{b^{3/2}}{400}\frac{3}{2}\frac{\eta}{27}(c_S - c_L - c_R) \leq \frac{c_R^{3/2}}{|R|} \leq \frac{(c_S - c_L)^{3/2}}{|S| - |L|}.$$

Proof. Let $\alpha = \frac{b^{3/2}}{400}\frac{3}{2}\frac{\eta}{27}$. The second inequality is obvious since $|R| = |S| - |L|$ and $c_S \geq c_L + c_R$. To prove the first inequality, let $\phi(x) = x^{3/2}/|R| - \alpha x$. Note that the derivative $\phi'(x) = (3/2)(x^{1/2}/|R|) - \alpha$ is increasing. Since $|R| \geq |S|/3$, we have $\phi'(c_S) \leq (1/2)(c_S^{1/2}/|S|) - \alpha$. Since S is internal, by definition of the algorithm $c_S \leq \kappa |S|^2$, which implies $\phi'(c_S) \leq 0$ by definition of κ and α. So, ϕ is decreasing in the range $[0, c_S]$. Since $c_R \leq c_S - c_L \leq c_S$, we deduce $\phi(c_R) \geq \phi(c_S - c_L)$, hence the lemma. □

Lemma 7.17. *For any $x \in [0,1]$ and $y \in [1/3, 2/3]$, we have:*

$$\frac{x^{3/2}}{y} + \frac{(1-x)^{3/2}}{1-y} \geq 4/3.$$

Proof. We study the minima of $f(x,y) = x^{3/2}/y + (1-x)^{3/2}/(1-y)$.

If x is at the boundary, say $x = 0$, then $f(x,y) = 1/(1-y)$ which has minimum $3/2$ for the specified range of y. The case $x = 1$ is symmetric. If x is not at a boundary then any local minimum must satisfy $\frac{\partial f}{\partial x} = 0$, hence $\frac{\sqrt{x}}{y} = \frac{\sqrt{1-x}}{1-y}$. If y is at the boundary, say $y = 1/3$, this gives $x = 1/5$, hence $f(x,y) = 3/\sqrt{5}$. The case $y = 2/3$ is symmetric. Finally, if (x,y) is in the interior of the domain then we must have $\frac{\partial f}{\partial y} = 0$ as well, hence $x = y = 1/2$, and $f(x,y) = \sqrt{2}$. Finally the minimum of those three values is $3/\sqrt{5} > 4/3$. □

Applying first Lemma 7.16 and then Lemma 7.17 for $x = c_L/c_S$ and $y = |L|/|S|$ yields

$$\begin{aligned}\frac{c_L^{3/2}}{|L|} + \frac{c_R^{3/2}}{|R|} &\geq \frac{c_L^{3/2}}{|L|} + \frac{(c_S - c_L)^{3/2}}{|S| - |L|} - \frac{b^{3/2}}{400}\frac{3}{2}\frac{\eta}{27}(c_S - c_L - c_R) \\ &\geq (4/3)\frac{c_S^{3/2}}{|S|} - \frac{b^{3/2}}{400}\frac{3}{2}\frac{\eta}{27}(c_S - c_L - c_R)\end{aligned}$$

Thus,

$$\frac{c_S^{3/2}}{|S|} \leq (3/4)\frac{c_L^{3/2}}{|L|} + (3/4)\frac{c_R^{3/2}}{|R|} + \frac{3}{4}\frac{b^{3/2}}{400}\frac{3}{2}\frac{\eta}{27}(c_S - c_L - c_R). \tag{7.7}$$

The rest of the analysis is straightforward. Applying Equation (7.7) from the root down, we bound the third sum of (7.6), call it A, as follows:

$$\begin{aligned} A &= \sum_{S \text{ internal}} \frac{400}{b^{3/2}} \frac{C(\pi_S^{loc})^{3/2}}{|S|} \\ &\leq \left(\sum_{S \text{ leaf}} \frac{400}{b^{3/2}} \frac{C(\pi_S^{loc})^{3/2}}{|S|} \sum_{i \geq 1} (3/4)^i \right) + \frac{3}{4} \frac{1}{400} \frac{3}{2} \frac{\eta}{27} C(\pi_V^{loc}). \end{aligned} \tag{7.8}$$

We bound the second term of (7.8) trivially by $(\eta/8)C(\pi_V^{in})$.

Let S be a leaf and S' its parent.[2] Since $C(\pi_S^{loc}) \leq C(\pi_{S'}^{loc})$ and $|S| \geq |S'|/3$, we have

$$\frac{C(\pi_S^{loc})^{1/2}}{|S|} \leq 3 \frac{C(\pi_{S'}^{loc})^{1/2}}{|S'|} \leq 3\sqrt{\kappa} = 3\sqrt{\frac{\eta^2 b^3}{350 \cdot 400^2}} \leq \frac{3\eta b^{3/2}}{18 \cdot 400}$$

by definition of internal nodes. Substituting and using the fact that leaves are disjoint, we bound the first term of (7.8) by

$$\sum_{S \text{ leaf}} \frac{400}{b^{3/2}} \frac{3\eta b^{3/2}}{18 \cdot 400} \frac{3/4}{1 - 3/4} C(\pi_S^{loc}) = \frac{\eta}{2} C(\pi_V^{loc}) \leq \frac{\eta}{2} C(\pi_V^{in}).$$

Therefore $A \leq (1/8 + 1/2)C(\pi_V^{in}) = 5/8 C(\pi_V^{in})$, completing the proof of the lemma.

7.5 Runtime

7.5.1 A faster algorithm

To speed up FAST-SCHEME, we perform the following two modifications: First, instead of calling IMPROVE just once with error tolerance $\epsilon b/5$, we first call it $\log(1/b)$ times with error tolerance $1/2$ before running it once with error tolerance $\epsilon/7$. The improved approximation scheme, denoted FASTER-SCHEME, is presented in Figure 7.5. Second, to bound the total runtime of the single-vertex moves, we need the cost to be monotone non-increasing. For that purpose, we modify algorithm IMPROVE, replacing the line

> Return IMPROVEREC(V, π)

with

> Return either π or IMPROVEREC(V, π), whichever has lower cost.

Lemma 7.7 clearly remains valid despite this modification.

We now prove that FASTER-SCHEME is a $1 + \epsilon$-approximation.

[2] If the root is a leaf no such parent exists, but in that case $A = 0$.

FASTER-SCHEME:
  Run KWIKSORT to define an ranking π
  for $i \leftarrow 1$ to $\lceil \log_2 1/b \rceil$ **do**
    $\pi \leftarrow$ IMPROVE$(\pi, 1/2)$
  end for
  Return IMPROVE$(\pi, \epsilon/7)$.

Figure 7.5: Faster approximation scheme (error tolerance $\epsilon > 0$)

Proof. (Of Theorem 7.2) Let $m = \lceil \log_2 1/b \rceil$ and π^i denote the ranking π after the i^{th} iteration of FASTER-SCHEME, $0 \leq i \leq m$. By the law of iterated expectations and Lemma 7.7, for any $1 \leq i \leq m$ we have

$$\mathbf{E}\left[C(\pi^i)\right] \quad = \quad \mathbf{E}\left[\mathbf{E}\left[C(\pi^i)|\pi^{i-1}\right]\right] \leq \mathbf{E}\left[OPT + \frac{1}{2}C(\pi^{i-1})\right] = OPT + \frac{1}{2}\mathbf{E}\left[C(\pi^{i-1})\right].$$

Therefore by Corollary 7.5 and the definition of m,

$$\mathbf{E}\left[C(\pi^m)\right] \quad \leq \quad 2OPT + 2^{-m}\frac{5}{b}OPT \leq 7OPT. \tag{7.9}$$

Finally, by Lemma 7.7, the expected cost of the output is at most

$$OPT + \frac{\epsilon}{7}\mathbf{E}\left[C(\pi^m)\right] = (1+\epsilon)OPT.$$

□

7.5.2 Analysis of running time

Here we analyze the runtime of FASTER-SCHEME. Throughout this section let $n = |V|$, the number of vertices in the overall input.

Lemma 7.18. *All the single vertex move local optimizations in* FASTER-SCHEME *have a combined expected runtime of* $O(n^3 \log n(\log(1/b) + \frac{1}{\epsilon}))$.

Proof. The modification to IMPROVE discussed in subsection 7.5.1 ensures that after a single vertex move yields a ranking with cost C, FASTER-SCHEME will never apply a single vertex move to an ranking costing at least C. Let π^0 denote the constant-factor ordering in FASTER-SCHEME. Each single vertex move multiplies the cost of the ranking by a factor of at most $1 - \frac{\eta}{4n\log_{3/2} n}$. Therefore after j moves, $C(\pi) \leq C(\pi^0)(1 - \frac{\eta}{4n\log_{3/2} n})^j \leq C(\pi^0)e^{-j\frac{\eta}{4n\log_{3/2} n}}$. Clearly $C(\pi) \geq OPT$ hence $\mathbf{E}[j]$ is at most $\mathbf{E}\left[\frac{4n\log_{3/2} n}{\eta}\ln\frac{OPT}{C(\pi^0)}\right] \leq \frac{4n\log_{3/2} n}{\eta}\ln\mathbf{E}\left[\frac{C(\pi^0)}{OPT}\right] \leq \frac{4n\log_{3/2} n}{\eta}\ln(5/b)$ moves are needed in expectation. During all the calls to IMPROVE except the last $\eta = 1/2$, hence at most $\frac{4n\log_{3/2} n}{1/2}\ln\frac{5}{b} = O\left(n\log n\log(1/b)\right)$ moves are required. During the final call to IMPROVE $\eta = \epsilon/7$ hence using a similar argument and (7.9) at most $\frac{4n\log_{3/2} n}{\epsilon/7}\ln\mathbf{E}\left[\frac{C(\pi^m)}{OPT}\right] = O\left(\frac{n\log_{3/2} n}{\epsilon}\right)$ moves are required. There are an additional $\lceil\log(1/b)\rceil + 1$ times that no improving move exists but this fact must still be verified. Overall one must check for local optimality and make an improving move if one exists $O(n\log n\log(1/b)) + O(\frac{n\log n}{\epsilon}) + O(\log 1/b) = O(n\log n(\log(1/b) + \frac{1}{\epsilon}))$ times.

Each check for local optimality can be trivially done in $O(n^2)$ time, so the single vertex moves take $O(n^3 \log n(\log(1/b) + \frac{1}{\epsilon}))$ time overall. □

Lemma 7.19. ImproveRec *has runtime* $O(n^2) + n2^{\tilde{O}(1/\eta^6)}$.

Proof. We refer to the call to ImproveRec made by Improve the *root call*. We first analyze the runtime of the AddApprox calls. Recall from Theorem 7.6 that each AddApprox call has runtime $O(|S|^2) + 2^{O(1/\eta^6)}$. Each vertex participates in exactly one AddApprox call, so the total runtime of the AddApprox calls descended from the root call is $O(n^2) + n2^{\tilde{O}(1/\eta^6)}$.

Each call to ImproveRec, excluding descendent ImproveRec and AddApprox calls, can easily be implemented to run in $O(|S|^2)$ time. Summing over the levels of the recursion tree gives total runtime $O\left(n^2 + n\frac{2n}{3} + n\frac{2^2n}{3^2} + \ldots\right) = O\left(n^2\right)$. □

We now prove the runtime portion of Theorem 7.2.

Proof. The overall runtime of FASTer-Scheme is:

$$
\begin{aligned}
& O(n \log n) && \text{(KwikSort Ailon and Mohri [2008])} \\
& + O(n^3 \log n(\log(1/b) + \tfrac{1}{\epsilon})) && \text{(Single vertex moves, Lemma 7.18)} \\
& + O(\log(1/b)) \left(O(n^2) + n2^{\tilde{O}(1/b^6)}\right) && \text{(ImproveRec } \eta = 1/2\text{, Lemma 7.19)} \\
& + \left(O(n^2) + n2^{\tilde{O}(1/(\epsilon b)^6)}\right) && \text{(ImproveRec } \eta = \epsilon/7\text{, Lemma 7.19)} \\
& = O\left(n^3 \log n(\log(\tfrac{1}{b}) + \tfrac{1}{\epsilon})\right) + n\left(2^{\tilde{O}(1/(\epsilon b)^6)}\right)
\end{aligned}
$$

□

7.6 Derandomization

The KwikSort algorithm was derandomized in [Van Zuylen et al. 2007; Van Zuylen and Williamson 2007]. The following theorem is implicit in the proof of Theorem 2.1 in Van Zuylen and Williamson [2007]:

Theorem 7.20. *[Van Zuylen and Williamson 2007] Let w be non-negative weight function on the edges. Assume that for every $u, v, x \in V$ with $w_{vu} - w_{uv} \leq \min\{w_{xv} - w_{vx}, w_{xu} - w_{ux}\}$ we have*

$$w_{vu} + \frac{1}{2}(w_{vx} + w_{xv}) + \frac{1}{2}(w_{xu} + w_{ux}) \geq \frac{1}{\alpha}(w_{uv} + w_{vx} + w_{xu})$$

for some $\alpha > 1$. Then there exists a deterministic polynomial-time α-approximation algorithm.

Theorem 7.21. *There exists a deterministic $3/b$-approximation algorithm for weighted feedback act set tournament.*

Proof. (Adapted from the proof by Van Zuylen and Williamson 2007 for the case $b = 1$.) We know $w_{vx} + w_{xv}, w_{xu} + w_{ux} \geq b$ and $w \leq 1$ hence

$$w_{vu} + \frac{1}{2}(w_{vx} + w_{xv}) + \frac{1}{2}(w_{xu} + w_{ux}) \geq 0 + \frac{b}{2} + \frac{b}{2} = b = \frac{1}{3/b}3 \geq \frac{1}{3/b}(w_{uv} + w_{vx} + w_{xu})$$

□

ImproveRec(vertices S, ranking π on S):
  if ImproveRec(S, π) previously computed **then**
    Return cached π^{out}.
  else if $|S| = 1$ **then**
    Return π
  else if $C(\pi) \geq \kappa |S|^2$ **then**
    Return the ranking from the deterministic additive error algorithm
    by Frieze and Kannan [1999] with $\delta = \frac{\eta}{4} \cdot \kappa$
  else
    Initialize π^{out} to an arbitrary ranking (e.g. π).
    for $k = \lceil |S|/3 \rceil$ to $\lfloor 2|S|/3 \rfloor$ **do**
      Let $L = \{ v \in S \ : \ \pi(v) \leq k \}$ and $R = S \setminus L$
      Let π_L be the ranking of L induced by π, i.e. $\pi_L(v) = \pi(v)$
      Let π_R be the ranking of R induced by π, i.e. $\pi_R(v) = \pi(v) - k$
      Let π^{temp} be the concat. of ImproveRec(L, π_L) and ImproveRec(R, π_R).
      if $C(\pi^{temp}) < C(\pi^{out})$ **then**
        $\pi^{out} \leftarrow \pi^{temp}$
      end if
    end for
    Return π^{out}.
  end if

Figure 7.6: Derandomized version of ImproveRec.

To derandomize FAST-Scheme we make three changes to our algorithms. Firstly, we replace AddApprox with the deterministic additive error algorithm by Frieze and Kannan [1999]. Secondly, we replace KwikSort with the deterministic constant-factor approximation algorithm of Theorem 7.21.

Thirdly we must eliminate the randomized choice of k in ImproveRec. We do this by trying every possible k and keeping the best ranking found. We cache intermediate results to prevent exponential blow-up in the runtime. The derandomized version of ImproveRec is given in Figure 7.6.

Let π^{loc} denote the input order. It is easy to see that every (S, π) pair encountered is of the form $S = \{ u \ : \ i \leq \pi^{loc}(u) \leq j \}$ and $\pi(u) = \pi^{loc}(v) - i$ for some $i \leq j$, so ImproveRec runs $O(n^2)$ times (excluding lookups in cache). Each call to ImproveRec has runtime dominated by the derandomized additive error algorithm, with runtime $n^{\tilde{O}(1/((\epsilon b)^3)^4)} = n^{\tilde{O}(1/(\epsilon b)^{12})}$. The overall runtime of Improve is therefore $O\left(n^2\right) \cdot n^{\tilde{O}(1/(\epsilon b)^{12})} = n^{\tilde{O}(1/(\epsilon b)^{12})}$.

The runtime of the derandomized FAST-Scheme is dominated by the $O(\log 1/b)$ calls to Improve so the overall runtime is $(\log 1/b) n^{\tilde{O}(1/(\epsilon b)^{12})} = n^{\tilde{O}(1/(\epsilon b)^{12})}$.

7.7 Appendix

In this appendix we prove Theorem 7.6. We want to find a $O(\delta n^2)$ additive approximation in time $O(n^2) + 2^{\tilde{O}(1/\delta^2)}$ for any constant $\delta > 0$.

Let a *bucketed ranking* be a function $\tilde{\pi}$ from V to $\{1, 2, 3, \ldots \lceil \frac{1}{\delta} \rceil\}$. If $\tilde{\pi}(v) = i$ we say that vertex

v is in bucket i.

Problem 7.22 (Bucketed FAS)**.**
Input: complete directed graph with vertex set V, $n \equiv |V|$ and non-negative edge weights w_{uv} with $w_{uv} + w_{vu} \leq 1$.
Output: A bucketed ranking minimizing $\tilde{C}(\tilde{\pi}) = \sum_{\{u,v\} \subset V : \tilde{\pi}(v) \geq \tilde{\pi}(u)} w_{vu}$.

Note the "$\geq$" in the objective function $\tilde{C}$ of Problem 7.22 – if vertices u and v are in the same bucket then $\tilde{C}$ pays for both w_{uv} and w_{vu}.

Now we prove Theorem 7.6.

Proof. Run Algorithm 2.2 and convert the output bucketed ranking $\widetilde{OUT}$ into a ranking OUT by concatenating the buckets and ordering arbitrarily within each bucket. The pessimistic definition of $\tilde{C}$ ensures $C(OUT) \leq \tilde{C}(\widetilde{OUT})$. On the other hand, the optimum ranking OPT can be converted into a bucketed ranking $\widetilde{OPT}$ that costs at most $O(\delta n^2)$ more by placing δn vertices in each bucket. Putting these remarks together with Theorem 2.2 we see

$$C(OUT) \quad \leq \quad \tilde{C}(\widetilde{OUT}) \leq \tilde{C}(\widetilde{OPT}) + O(\delta n^2) \leq C(OPT) + O(\delta n^2) + O(\delta n^2)$$

proving the approximation factor portion of Theorem 7.6.

Determining where to place each vertex can be done in time $O(m/\delta)$ totally naïvely, where m is the number of vertices placed so far. Simple incremental data structures allow this to be reduced to $O(m + \frac{1}{\delta})$, which yields the runtime $O(n^2) + 2^{\tilde{O}(1/\delta^2)}$ of Theorem 7.6. □

Chapter 8

Ranking MIN-CSPs: approximation algorithms

8.1 Introduction

The results presented in this chapter are joint work with Marek Karpinski. They previously appreared in Karpinski and Schudy [2009a].

We study the approximability of the Minimum Betweenness problem in tournaments (see Ailon and Alon [2007]) that resisted so far efforts of designing polynomial time approximation algorithms with a constant approximation ratio. For the status of the general Betweenness problem, see e.g. Opatrny [1979], Chor and Sudan [1998], Ailon and Alon [2007], Charikar et al. [2009].

In this chapter we design the first *polynomial time approximation scheme* (PTAS) for that problem, and generalize it to much more general class of ranking CSP problems, called here *fragile* problems. To our knowledge it is the first nontrivial approximation algorithm for the Betweenness problem in tournaments.

In the Betweenness problem we are given a ground set of *vertices* and a set of *betweenness constraints* involving 3 vertices and a *designated* vertex among them. The objective function of a ranking of the elements is the number of betweenness constraints for which the designated vertex is not between the other two vertices. The goal is to *minimize* the objective function. We refer to the Betweenness problem in tournaments, that is in instances with a constraint for every triple of vertices, as the BetweennessTour or *fully dense* Betweenness problem (see Ailon and Alon [2007]). We consider also the k-ary extension k-FAST of the Feedback Arc Set in tournaments (FAST) problem (see Chapter 7).

We extend the above classes by introducing a more general class of *fragile ranking* k-CSP problems. A *constraint* S of a ranking k-CSP problem is called *fragile* if changing the relative order of a single vertex in S with respect to the rest of S makes it *unsatisfied* whenever S was satisfied by the

original order. A *ranking* k-CSP problem is called *fragile* if all its constraints are fragile.

We now formulate our main results.

Theorem 8.1. *There exists a PTAS for the* BetweennessTour *problem.*

The above answers an open problem of Ailon and Alon [2007] on the approximation status of the Betweenness problem in tournaments.

We now formulate our first generalization.

Theorem 8.2. *There exist PTASs for all* fragile ranking k*-CSP problems in tournaments.*

Theorem 8.2 entails, among other things, existence of a PTAS for the k-ary extension of FAST.

Corollary 8.3. *There exists a PTAS for the* k-FAST *problem.*

We generalize BetweennessTour to arities $k \geq 4$ by specifying for each constraint S a pair of vertices in S that must be placed at the ends of the ranking induced by the vertices in S. Such constraints do not satisfy our definition of fragile, but do satisfy a weaker notion that we call *weak fragility*. The definition of weakly fragile is identical to the definition for fragile except that only four particular single vertex moves are considered, namely swapping the first two vertices, swapping the last two, and moving the first or last vertex to the other end. We now formulate our most general theorem.

Theorem 8.4. *There exist PTASs for all* weak-fragile ranking k*-CSP problems in tournaments.*

Corollary 8.5. *There exists a PTAS for the* k-BetweennessTour *problem.*

All our PTASs are randomized but one can easily derandomize them by exhaustively considering every possible random choice.

We give the algorithms and the analysis of our PTAS in Sections 8.3-8.8.

Betweenness and generalizations

The betweenness problem has input consisting of information of the form object b must be between objects a and c in the ordering. It is NP hard in general to determine if there is an ordering consistent with all of the input [Chor and Sudan, 1998, Opatrny, 1979]. This implies that it is NP-hard to approximate the problem of minimizing disagreements to any factor. The problem of maximizing agreements has a constant factor approximation algorithm and no PTAS unless P=NP [Chor and Sudan, 1998].

We study the problem of minimizing disagreements in the special case of instances with information available for every set of three objects (fully dense). An easy reduction from FAST shows this problem is still NP-hard. Ailon and Alon [Ailon and Alon, 2007] show that in fact fully dense betweenness is NP-hard to approximate better than an additive $O(n^{2-\epsilon})$ for any $\epsilon > 0$. We give a PTAS for this problem, which to our knowledge is the first non-trivial approximation algorithm for it.

As described in Section 8.3 we generalize our PTAS to a variety of fully dense ranking problems.

8.2 Intuition and main ideas

Our first key idea is analogous to the approximation of a differentiable function by a tangent line. Given a ranking π and any ranking CSP, the change in cost from switching to a similar ranking π' can be well approximated by the change in cost of a particular weighted feedback arc set problem (see proof of Lemma 8.22). Furthermore if the ranking CSP is fragile and fully dense the corresponding feedback arc set instance is a (weighted) tournament (Lemma 8.16). So *if* we somehow had access to a ranking similar to the optimum ranking π^* we could create this FAST instance and run the PTAS for FAST described in Chapter 7 to get a good ranking.

We do not have access to π^* but we can use a variant of the fragile techniques from Chapter 3 to get close. We pick a random sample of vertices and guess their location in the optimal ranking to within ϵn. We then create an ordering σ^1 greedily from the random sample. We show that this ordering is close to π^*, in that $|\pi^*(v) - \sigma^1(v)| = O(\epsilon n)$ for all but $O(\epsilon n)$ of the vertices (Lemma 8.11).

We then do a second greedy step (relative to σ^1), creating σ^2. We then identify a set U of *unambiguous* vertices for which we know $|\pi^*(v) - \sigma^2(v)| = O(\epsilon n)$ (Lemma 8.15). We temporarily set aside the $O(OPT/(\epsilon n^{k-1}))$ (Lemma 8.14) remaining vertices. These two greedy steps are similar in spirit to previous work on ordinary (non-ranking) everywhere-dense fragile CSPs described in Chapter 3 but substantially more involved.

We then use σ^2 to create a FAST instance w that locally represents the CSP. Unfortunately the error in σ^2 causes the weights of w to have significant error (Lemma 8.18) even when $OPT \approx 0$. At first glance even an exact solution to this FAST problem would seem insufficient, for how can solving a problem similar to the desired one lead to a precisely correct solution? We show that FAST is tolerant of such errors (Lemma 8.22). The intuition for why this is possible is that minor adjustments to edge weights of a zero-cost FAST instance change the optimum *cost* but leave the optimum *ranking* unchanged.

Another difficulty is that the incorrect weights in FAST instance w may increase the optimum cost of w far above OPT, leaving the PTAS for FAST free to return a poor ranking. To remedy this we create a new FAST instance $\bar{w}$ by canceling weight on opposing edges, i.e. reducing w_{uv} and w_{vu} by the same amount. The resulting simplified instance $\bar{w}$ clearly has the same optimum ranking as w but a smaller optimum value. The PTAS for FAST requires that the ratio of the maximum and the minimum of $w_{uv} + w_{vu}$ must be bounded above by a constant so we limit the amount of cancellation to ensure this (Lemma 8.16). It turns out that this cancellation trick is sufficient to ensure that the PTAS for FAST does not introduce too much error (Lemma 8.19).

Finally we greedily insert the relatively few ambiguous vertices into the ranking output by the PTAS for FAST.

8.3 Approximation Algorithm

First we state some core notation. Throughout this paper let V refer to the set of objects (vertices) being ranked and n denotes $|V|$. Our $O(\cdot)$ hides k but not ϵ or n. Our $\tilde{O}(\cdot)$ hides $(\log(1/\epsilon))^{O(1)}$. A *ranking* is a bijective mapping from a ground set $S \subseteq V$ to $\{1, 2, 3, \ldots, |S|\}$. An *ordering* is an injection from S into $\mathbb{R}$. We use π and σ (plus superscripts) to denote orderings and rankings respectively. Let π^* denote an optimal ordering and OPT its cost. We let $\binom{n}{k}$ (for example) denote the standard binomial coefficient and $\binom{V}{k}$ denote the set of subsets of set V of size k.

For any ordering σ let $Ranking(\sigma)$ denote the ranking naturally associated with σ. To help prevent ties we relabel the vertices so that $V = \{1, 2, 3, \ldots, |V|\}$. We will often choose to place u in one of $O(1/\epsilon)$ positions $\mathcal{P}(u) = \{j\epsilon n + u/(n+1), 0 \leq j \leq 1/\epsilon\}$ (the $u/(n+1)$ term breaks ties), where $\epsilon > 0$ is the desired approximation parameter. We say that an ordering is a *bucketed ordering* if $\sigma(u) \in \mathcal{P}(u)$ for all u. Let $Round(\pi)$ denote the bucketed ordering corresponding to π (rounding down), i.e. $Round(\pi)(u)$ equals $\pi(u)$ rounded down to the nearest multiple of ϵn, plus $u/(n+1)$.

Let $v \mapsto p$ denote the ordering over $\{v\}$ which maps v to p. For set Q of vertices and ordering σ with domain including Q let $Q \mapsto \sigma$ denote the ordering over Q which maps $u \in Q$ to $\sigma(u)$, i.e. the restriction of σ to Q. For orderings σ^1 and σ^2 with disjoint domains let $\sigma^1 \big| \sigma^2$ denote the natural combined ordering over $Domain(\sigma^1) \cup Domain(\sigma^2)$. For example of our notations, $Q \mapsto \sigma \big| v \mapsto p$ denotes the ordering over $Q \cup \{v\}$ that maps v to p and $u \in Q$ to $\sigma(u)$.

A ranking k-CSP consists of a ground set V of *vertices*, an arity $k \geq 2$, and a *constraint system* c, where c is a function from rankings of k vertices to $\{0, 1\}$.[1] We say that a subset $S \subset V$ of size k is *satisfied* in ordering σ of S if $c(Ranking(\sigma)) = 0$. For brevity we henceforth abuse notation and omit the "$Ranking$" and write simply $c(\sigma)$. The objective of a ranking CSP is to find an ordering σ (w.l.o.g. a ranking) minimizing the number of unsatisfied constraints, which we denote by $C^c(\sigma) = \sum_{S \in \binom{Domain(\sigma)}{k}} c(S \mapsto \sigma)$. We will frequently omit the superscript c, in which case it should be understood to be the constraint system of the overall problem we are trying to solve.

Abusing notation we sometimes refer to $S \subseteq V$ as a *constraint*, when we really are referring to $c(S \mapsto \cdot)$. A constraint S is *fragile* if whenever it is satisfied making any single vertex move that changes the relative order of the vertices in S makes it unsatisfied. In other words constraint S is fragile if $c(S \to \pi) + c(S \to \pi') \geq 1$ for all rankings π and π' over S that differ by a single vertex move, i.e. $\pi' = Ranking(v \to p \big| S \setminus \{v\} \to \pi)$ for some $v \in S$ and $p \in (\mathbb{Z} + 1/2)$. Fragility is illustrated in Figure 8.1.

A constraint S is *weakly fragile* if $c(S \to \pi) + c(S \to \pi') \geq 1$ for all rankings π and π' that differ by a swap of the first two vertices, the last two, or cyclic shift of a single vertex. In other words $\pi' = Ranking(v \to p \big| S \setminus \{v\} \to \pi)$ for some $v \in S$ and $p \in \mathbb{R}$ with $(\pi(v), p) \in \{(1, 2 + \frac{1}{2}), (1, k + \frac{1}{2}), (k, k - \frac{3}{2}), (k, \frac{1}{2})\}$. Observe that this is equivalent to ordinary fragility for $k \leq 3$. Weak fragility is illustrated in Figure 8.2

Our techniques handle ranking CSPs that are *fully dense* with weakly fragile constraints, i.e.

[1] Our results transparently generalize to the $[0, 1]$ case as well, but the 0/1 case allows simpler terminology.

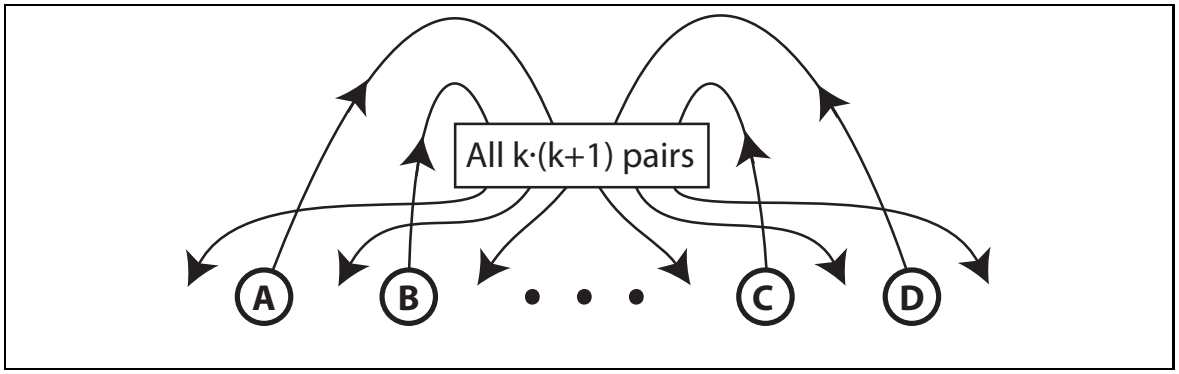

Figure 8.1: An illustration of fragility. For a constraint to be fragile all the illustrated single vertex moves must make any satisfied constraint unsatisfied.

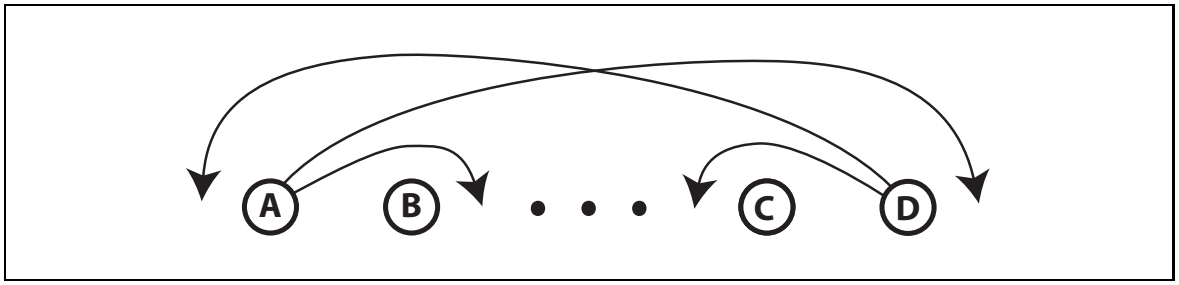

Figure 8.2: An illustration of weak fragility. For a constraint to be weak fragile all the illustrated single vertex moves must make any satisfied constraint unsatisfied.

every set S of k vertices corresponds to a weakly fragile constraint. Fully dense instances are also known as tournaments.

Let $b^c(\sigma, v, p) = \sum_{Q:\dots} c(Q \mapsto \sigma\big|v \mapsto p)$, where the sum is over sets $Q \subseteq \mathit{Domain}(\sigma) \setminus \{v\}$ of size $k-1$. Note that this definition is valid regardless of whether or not v is in $\mathit{Domain}(\sigma)$. The only requirement is that the range of σ excluding $\sigma(v)$ must not contain p. This ensures that the argument to $c(\cdot)$ is an ordering (injective). We will usually omit the superscript c (as with C).

We call a non-negative weight function w over the edges of the complete graph induced by some vertex set U a *FAS instance*. We can express the FAST problem in our framework by the correspondence $c(u \mapsto x\big|v \mapsto y) = \begin{cases} w_{vu} & \text{if } x < y \\ w_{uv} & \text{otherwise} \end{cases}$. Abusing notation slightly we also write $C^w(\sigma)$ for $C^c(\sigma)$ with the above c. More concretely $C^w(\sigma) = \sum_{u,v:\sigma(u)>\sigma(v)} w_{uv}$. Similarly we write $b^w(\sigma, v, p) = \sum_{u \neq v} \begin{cases} w_{uv} & \text{if } \sigma(u) > p \\ w_{vu} & \text{if } \sigma(u) < p \end{cases}$. Observe that FAST captures all possible fragile constraints with $k = 2$. We generalize to k-FAST as follows: a k-FAST constraint over S is satisfied by one particular ranking of S and no others.

We generalize BETWEENNESSTOUR to $k \geq 4$ as follows. Each constraint S designates two vertices $\{u, v\}$, which must be the first and last positions, i.e. if π is the ranking of the vertices in S then $c(\pi) = \mathbb{1}(\{\pi(u), \pi(v)\} \neq \{1, k\})$.

Theorem 2.2 from Chapter 2 entails the following corollary.

Input: Vertex set V, $|V| = n$, arity k, system c of fully dense arity k constraints, and approximation parameter $\epsilon > 0$.

1: Run AddApprox($\epsilon^5 n^k$) and return the result if its cost is at least $\epsilon^4 n^k$

2: Pick sets $T_1, \ldots, T_t$ uniformly at random with replacement from $\binom{V}{k-1}$, where $t = \frac{14 \ln(40/\epsilon)}{\binom{k}{2}\epsilon}$. Guess (by exhaustion) bucketed ordering σ^0, which is the restriction of $Round(\pi^*)$ to the sampled vertices $\bigcup_i T_i$.

3: Compute bucketed ordering σ^1 greedily with respect to the random samples and σ^0: $\sigma^1(u) = \text{argmin}_{p \in \mathcal{P}(u)} \hat{b}(u,p)$ where $\hat{b}(u,p) = \frac{\binom{n}{k-1}}{t} \sum_{i: u \notin T_i} c(T_i \mapsto \sigma^0 \big| v \mapsto p)$.

4: For each vertex v: If $b(\sigma^1, v, p) \leq 13k^4 3^{k-1}\epsilon\binom{n-1}{k-1}$ for some $p \in \mathcal{P}(v)$ then call v *unambiguous* and set $\sigma^2(v)$ to the corresponding p (pick any if multiple p satisfy). Let U denote the set of unambiguous vertices, which is the domain of bucketed ordering σ^2.

5: Compute feedback arc set instance over unambiguous vertices U with weights $\bar{w}^{\sigma^2}_{uv}$ (see text). Solve it using FAST PTAS. Do single vertex moves until local optimality (with respect to FAST objective function), yielding ranking π^3 of U.

6: Create ordering σ^4 over V defined by $\sigma^4(u) = \begin{cases} \pi^3(u) & \text{if } u \in U \\ \text{argmin}_{p=v/(n+1)+j, 0 \leq j \leq n} b(\pi^3, u, p) & \text{otherwise} \end{cases}$ In other words insert each vertex $v \in V \setminus U$ into $\pi^3(v)$ greedily.

7: Return $\pi^4 = Ranking(\sigma^4)$.

Figure 8.3: A $1 + O(\epsilon)$-approximation for weak fragile rank k-CSPs in tournaments.

Corollary 8.6. *For any $\delta > 0$ and constraint system c there is an algorithm* AddApprox *for the problem of finding a ranking π with $C(\pi) \leq C(\pi^*) + \delta n^k$. Its runtime is $n^{O(1)} 2^{\tilde{O}(1/\delta^2)}$.*

For any ordering σ with domain U let w^σ_{uv} equal the number of the constraints $\{u, v\} \subseteq S \subseteq U$ with $c(\sigma') = 1$ where (1) $\sigma' = S \setminus \{v\} \mapsto \sigma \big| v \mapsto p$, (2) $p = \sigma(u) - \delta$ if $\sigma(v) > \sigma(u)$ and $p = \sigma(v)$ otherwise, and (3) $\delta > 0$ is sufficiently small to put p adjacent to $\sigma(u)$. In other words if v is after u in σ it is placed immediately before v in σ'. Observe that $0 \leq w_{uv} \leq \binom{|U|-2}{k-2}$. We use the abbreviation $C^{\sigma'}(\sigma) = C^{w^{\sigma'}}(\sigma)$. The following Lemma follows easily from the definitions.

Lemma 8.7. *For any ordering σ we have (1) $C^\sigma(\sigma) = \binom{k}{2} C(\sigma)$ and (2) $b^{w^\sigma}(\sigma, v, \sigma(v)) = (k-1) \cdot b(\sigma, v, \sigma(v))$ for all v.*

Proof. Observe that all w_{uv} that contribute to $C^\sigma(\sigma)$ or $b^{w^\sigma}(\sigma, v, \sigma(v))$ satisfy $\sigma(u) > \sigma(v)$ and hence such w_{uv} are equal to the number of constraints containing u and v that are unsatisfied in σ. The $\binom{k}{2}$ and $k-1$ factors appear because constraints are counted multiple times. □

We define $\bar{w}^\sigma_{uv} = w^\sigma_{uv} - \min(\frac{1}{10 \cdot 3^{k-1}}\binom{n-2}{k-2}, w^\sigma_{uv}, w^\sigma_{vu})$, where U is the domain of σ. Let $\bar{C}^\sigma(\sigma') = C^{\bar{w}^\sigma}(\sigma')$. Observe that w and $\bar{w}$ are equivalent from an exact solution point of view, but $\bar{w}$ has a smaller objective value for approximation purposes. In other words $C^\sigma(\pi') - C^\sigma(\pi^\circ) = \bar{C}^\sigma(\pi') - \bar{C}^\sigma(\pi^\circ)$ for all rankings π' and π°.

For any orderings σ and σ' with domain U, we say that $\{u, v\} \subseteq U$ is a *σ/σ'-inversion* if $\sigma(u) - \sigma(v)$ and $\sigma'(u) - \sigma'(v)$ have different signs. Let $d(\sigma, \sigma')$ denote the number of σ/σ'-inversions (a.k.a. Kendall Tau distance). We say that v does a *left to right (σ, p, σ', p')-crossing* if $\sigma(v) < p$ and $\sigma'(v) > p'$. We say that v does a *right to left $(\sigma, p/\sigma', p')$-crossing* if $\sigma(v) > p$ and $\sigma'(v) < p'$. We

say that v does a (σ, p, σ', p')-*crossing* if v does a crossing of either sort. We say that u σ/σ'-*crosses* $p \in \mathbb{R}$ if it does a (σ, p, σ', p)-crossing.

With these notations in hand we now formalize the ideas described in Section 8.2 in our Algorithm 8.3. The non-deterministic "guess (by exhaustive sampling)" on line 2 of our algorithm should be implemented in the traditional manner: place the remainder of the algorithm in a loop over possible orderings of the sample, with the overall return value equal to the best of the π^4 rankings found. Our algorithm can be derandomized by choosing $T_1, \ldots, T_t$ non-deterministically rather than randomly; see Section 8.4 for details.

If $OPT \geq \epsilon^4 n^k$ then the first line of the algorithm is sufficient for a PTAS so for the remainder of the analysis we assume that $OPT \leq \epsilon^4 n^k$. For most of the analysis we actually need something weaker, namely that OPT is at most some sufficiently small constant times $\epsilon^2 n^k$. We only need the full $OPT \leq \epsilon^4 n^k$ in one place in Section 8.8.

8.4 Runtime analysis

By Theorem 8.6 the additive approximation step takes time $n^{O(1)} 2^{\tilde{O}(1/\epsilon^{10})}$. There are at most $(1/\epsilon)^{t\cdot(k-1)} = 2^{\tilde{O}(1/\epsilon)}$ bucketed orderings σ^0 to try. The PTAS for FAST takes time $n^{O(1)} 2^{\tilde{O}(1/\epsilon^6)}$ by Theorem 7.2. The overall runtime is

$$n^{O(1)} 2^{\tilde{O}(1/\epsilon^{10})} + 2^{\tilde{O}(1/\epsilon)} \cdot \left(n^{O(1)} + n^{O(1)} 2^{\tilde{O}(1/\epsilon^6)} \right) = n^{O(1)} 2^{\tilde{O}(1/\epsilon^{10})}.$$

Derandomization increases the runtime of the two algorithms that we use as subroutines to $n^{poly(1/\epsilon)}$. There are at most $n^{t\cdot(k-1)} = n^{\tilde{O}(1/\epsilon)}$ possible sets $T_1, \ldots T_t$ that the derandomized algorithm must consider. Therefore the overall runtime is

$$(n^{poly(1/\epsilon)} + n^{poly(1/\epsilon)} \cdot 2^{\tilde{O}(1/\epsilon)} \cdot n^{poly(1/\epsilon)} = n^{poly(1/\epsilon)}.$$

8.5 Analysis of σ^1

Let $\sigma^\square = \mathit{Round}(\pi^*)$. Call vertex v *costly* if $b(\sigma^\square, v, \sigma^\square(v)) \geq 2\binom{k}{2}\epsilon\binom{n-1}{k-1}$ and *non-costly* otherwise.

Lemma 8.8. *The number of costly vertices is at most* $\frac{k\cdot OPT}{\epsilon\binom{k}{2}\binom{n-1}{k-1}}$.

Proof. First observe that for any costly v we have

$$2\binom{k}{2}\epsilon\binom{n-1}{k-1} \leq b(\sigma^\square, v, \sigma^\square(v)) \leq b(\pi^*, v, \pi^*(v)) + \epsilon\binom{k}{2} \cdot \binom{n-1}{k-1}$$

since only at most a $\epsilon\binom{k}{2}$ fraction of the $\binom{n-1}{k-1}$ possible constraints contain a $\pi^*/\sigma^\square$-inversion. Therefore

$$b(\pi^*, v, \pi^*(v)) \geq 2\binom{k}{2}\epsilon\binom{n-1}{k-1} - \epsilon\binom{k}{2} \cdot \binom{n-1}{k-1} = \epsilon\binom{k}{2} \cdot \binom{n-1}{k-1}$$

Secondly observe that $kC(\pi^*) = \sum_v b(\pi^*, v, \pi^*(v)) \geq (\text{number costly})\epsilon\binom{k}{2}\binom{n-1}{k-1}$, completing the proof. ∎

Lemma 8.9. *Let σ be an ordering of V, $|V| = n$, $v \in V$ be a vertex and $p, p' \in \mathbb{R}$. Let B be the set of vertices (excluding v) between p and p' in σ. Then $b(\sigma, v, p) + b(\sigma, v, p') \geq \frac{|B|}{(n-1)3^{k-1}}\binom{n-1}{k-1}$.*

Proof. By definition

$$b(\sigma, v, p) + b(\sigma, v, p') = \sum_{Q:\cdots} [c(Q \mapsto \sigma \big| v \mapsto p) + c(Q \mapsto \sigma \big| v \mapsto p')] \tag{8.1}$$

where the sum is over sets $Q \subseteq U \setminus \{v\}$ of $k-1$ vertices. Observe by weak fragility that the quantity in brackets in (8.1) is at least 1 for every Q that either has all $k-1$ vertices between p and p' in σ^2 or has one vertex between them and the remaining $k-2$ either all before or all after.

We consider two cases. If $|B| \geq |V|/3$ then the number of such Q is at least $\binom{|B|}{k-1} = \frac{|B|}{k-1}\binom{|B|-1}{k-2} \geq \frac{|B|}{2\cdot(k-1)3^{k-2}}\binom{n-2}{k-2}$ for sufficiently large n. If $|B| < |V|/3$ then either at least $|V|/3$ vertices are before or at least $|V|/3$ vertices are after hence the number of such Q is at least $|B|\binom{|V|/3}{k-2} \geq \frac{|B|}{2\cdot 3^{k-2}}\binom{n-2}{k-2} \geq \frac{|B|}{(k-1)\cdot 3^{k-1}}\binom{n-2}{k-2}$ for sufficiently large n. ☐

For vertex v we say that a position $p \in \mathcal{P}(v)$ is *v-out of place* if there are at least $6\binom{k}{2}3^{k-1}\epsilon n$ vertices between p and $\sigma^{\square}(v)$ in $\sigma^{\square}$. We say vertex v is *out of place* if $\sigma^1(v)$ is v-out of place.

Lemma 8.10. *The number of non-costly out of place vertices is at most $\epsilon n/2$ with probability at least 9/10.*

Proof. Focus on some $v \in V$ and $p \in \mathcal{P}(v)$. From the definition of out-of-place and Lemma 8.9 we have

$$b(\sigma^{\square}, v, \sigma^{\square}) + b(\sigma^{\square}, v, p) \geq \frac{6\binom{k}{2}3^{k-1}\epsilon n}{(n-1)3^{k-1}}\binom{n-1}{k-1} \geq 6\epsilon\binom{k}{2}\binom{n-1}{k-1}$$

for any v-out of place p. Next recall that for costly v we have

$$b(\sigma^{\square}, v, \sigma^{\square}(v)) \leq 2\binom{k}{2}\epsilon\binom{n-1}{k-1} \tag{8.2}$$

hence

$$b(\sigma^{\square}, v, p) \geq 4\binom{k}{2}\epsilon\binom{n-1}{k-1} \tag{8.3}$$

for any v-out of place p.

Recall that

$$\hat{b}(v, p) = \frac{\binom{n}{k-1}}{t}\sum_{i: v \notin T_i} c(T_i \mapsto \sigma^0 \big| v \mapsto p)$$

for any p. Each term of the sum is a 0/1 random variable with mean $\mu(p) = \frac{1}{\binom{n}{k-1}}\sum_{Q \in \binom{V}{k-1}: v \notin Q} c(Q \mapsto \sigma^{\square}\big| v \mapsto p) = \frac{1}{\binom{n}{k-1}} b(\sigma^{\square}, v, p)$. Therefore $\mathbf{E}\left[\hat{b}(v, p)\right] = b(\sigma^{\square}, v, p)$. We can bound $\mu(\sigma^{\square}(v)) \leq 2\binom{k}{2}\epsilon\binom{n-1}{k-1}/\binom{n}{k-1} \equiv M$ using (8.2). For any v-out of place p we can bound $\mu(p) \geq 2M$ by (8.3).

We can bound the probability that sum in $\hat{b}(v, \sigma^{\square}(v))$ is at least $(1 + 1/3)Mt$ using a Chernoff bound as

$$\exp(-(1/3)^2 Mt/3) \leq \exp\left(-\frac{1}{9}\cdot\frac{1}{\binom{n}{k-1}}\cdot 2\binom{k}{2}\epsilon\binom{n-1}{k-1}\cdot\frac{14\ln(40/\epsilon)}{\binom{k}{2}\epsilon}\cdot\frac{1}{3}\right) \leq \epsilon/40$$

for sufficiently large n. Similarly for any v-out of place p we can bound the probability that $\hat{b}(v,p)$ is at most $(1-1/3)Mt$ by $\exp(-(1/3)^2 Mt/2) \leq (\epsilon/40)^3$. Therefore by union bound the probability of some v-out of place p having $\hat{b}(v,p)$ too small is at most $\epsilon^2/40^3 \leq \epsilon/40$. Clearly $4(1-1/3) \geq 2(1+1/3)$ so each vertex v is out of place with probability at least $\epsilon/20$. A Markov bound completes the proof. □

Lemma 8.11. *With probability at least 9/10 we have*

1. *The number of out of place vertices is at most ϵn.*
2. *The number of vertices v with $|\sigma^1(v) - \sigma^{\square}(v)| > 3k^2 3^{k-1}\epsilon n$ is at most ϵn*
3. *$d(\sigma^1, \sigma^{\square}) \leq 6k^2 3^{k-1}\epsilon n^2$*

Proof. By Lemma 8.8 and the fact $OPT \leq \epsilon^4 n^k$ we have at most $\frac{k \cdot OPT}{\binom{k}{2}\epsilon\binom{n-1}{k-1}} \leq \epsilon n/2$ costly vertices for n sufficiently large. Therefore Lemma 8.10 implies the first part of the Lemma.

Observe that any vertex with $|\sigma^1(v) - \sigma^{\square}(v)| > 3k^2\epsilon n \geq (6\binom{k}{2}+1)\epsilon n$ must necessarily be v-out of place, completing the proof of the second part of the Lemma.

For the final part observe that if u and v are a $\sigma^1/\sigma^{\square}$-inversion and not among the ϵn out of place vertices then there can be at most $2 \cdot 6\binom{k}{2}3^{k-1}\epsilon n$ vertices between $\sigma^{\square}(v)$ and $\sigma^{\square}(u)$ in $\sigma^{\square}$. Each u therefore only $24\binom{k}{2}3^{k-1}\epsilon n$ possibilities for v. Therefore $d(\sigma^1, \sigma^{\square}) \leq \epsilon n^2 + 24\binom{k}{2}3^{k-1}\epsilon n \cdot n/2 \leq 6\epsilon k^2 3^{k-1} n^2$. □

Our remaining analysis is deterministic, conditioned on the event of Lemma 8.11 holding.

8.6 Analysis of σ^2

The following key Lemma shows the sensitivity of $b(\sigma, v, p)$ to its first and third arguments.

Lemma 8.12. *For any constraint system c with arity $k \geq 2$, orderings σ and σ' over vertex set $T \subseteq V$, vertex $v \in V$ and $p, p' \in \mathbb{R}$ we have*

1. $$|b^c(\sigma, v, p) - b^c(\sigma', v, p')| \leq \binom{n-2}{k-2}(\textit{number of crossings}) + \binom{n-3}{k-3}d(\sigma, \sigma')$$
2. $$|b^c(\sigma, v, p) - b^c(\sigma', v, p')| \leq \binom{n-2}{k-2}\left(|\textit{net flow}| + k\sqrt{d(\sigma, \sigma')}\right)$$

where $\binom{n-3}{k-3} = 0$ if $k = 2$, (net flow) is $|\{\, v \in T \;:\; \sigma'(v) > p' \,\}| - |\{\, v \in T \;:\; \sigma(v) > p \,\}|$, and (number of crossings) is the number of $v \in T$ that do a (σ, p, σ', p')-crossing.

Proof. Fix σ, σ', T, v, p and p'. Let L (resp. R) denote the vertices in T that do left to right (resp. right to left) (σ, p, σ', p')-crossings. It is easy to see that a constraint $\{v\} \cup Q$, $Q \in \binom{T \setminus \{v\}}{k-1}$ contributes identically to $b(\sigma, v, p)$ and $b(\sigma', v, p')$ unless either:

1. Q and $(L \cup R)$ have non-empty intersection (or)

2. Q contains a σ/σ'-inversion $\{s,t\}$.

The first part of the Lemma follows easily.

Towards proving the second part we first bound $|L|+|R|$. Observe that $|L| = |R| + (net\ flow)$. Assume w.l.o.g. that $(net\ flow) \geq 0$. Observe that every pair $v \in L$ and $w \in R$ are a σ/σ'-inversion, hence $d(\sigma,\sigma') \geq |L| \cdot |R| = (|R| + (net\ flow))|R| \geq |R|^2$. We conclude that $|L|+|R| = 2|R| + (net\ flow) \leq 2\sqrt{d(\sigma,\sigma')} + (net\ flow)$. Therefore the number of constraints of the first type is at most $\binom{n-2}{k-2}(2\sqrt{d(\sigma,\sigma')} + (net\ flow))$.

To simplify we bound

$$\begin{aligned}\binom{n-3}{k-3} d(\sigma,\sigma') &= \binom{n-2}{k-2}\sqrt{d(\sigma,\sigma')} \cdot \frac{k-2}{n-2} \cdot \sqrt{d(\sigma,\sigma')} \\ &\leq \binom{n-2}{k-2}\sqrt{d(\sigma,\sigma')} \cdot (k-2)\frac{\sqrt{n(n-1)/2}}{n-2} \leq (k-2)\binom{n-2}{k-2}\sqrt{d(\sigma,\sigma')}\end{aligned}$$

for sufficiently large n. □

Observe that the quantity *net flow* in Lemma 8.12 is zero whenever $p = p'$ and σ and σ' are both *rankings*. Therefore we have the following useful corollary.

Corollary 8.13. *Let π and π' be rankings over vertex set U and w a FAST instance over U. Then $|b^w(\pi,v,p) - b^w(\pi',v,p)| \leq 2(\max_{r,s} w_{rs})\sqrt{d(\pi,\pi')}$ for all v and $p \in \mathbb{R} \setminus \mathbb{Z}$.*

Lemma 8.14. *For U in Algorithm 8.3 we have $|V \setminus U| \leq \frac{k \cdot OPT}{\epsilon\binom{k}{2}\binom{n-1}{k-1}} = O(\frac{n}{\epsilon} \cdot \frac{OPT}{n^k})$.*

Proof. Observe that the number of vertices that $\sigma^\square/\sigma^1$-cross a particular p is at most $2 \cdot 6k^2 3^{k-1}\epsilon n$ by Lemma 8.11 (first part). Therefore we apply Lemmas 8.11 and 8.12, yielding

$$|b(\sigma^\square,v,p) - b(\sigma^1,v,p)| \leq \binom{n-2}{k-2} 12k^2 3^{k-1}\epsilon n + \binom{n-3}{k-3} 6k^2 3^{k-1}\epsilon n^2 \leq 12\epsilon k^4 3^{k-1}\binom{n-1}{k-1} \quad (8.4)$$

for all v and p.

Fix a non-costly v. By definition of costly $b(\sigma^\square, v, \sigma^\square(v)) \leq 2\binom{k}{2}\epsilon\binom{n-1}{k-1} \leq k^4 3^{k-1}\epsilon\binom{n-1}{k-1}$, hence $b(\sigma^1, v, \sigma^\square(v)) \leq 13k^4 3^{k-1}\epsilon\binom{n-1}{k-1}$, so $v \in U$.

Finally recall Lemma 8.8. □

We define $\pi^\circledast$ to be the ranking induced by the restriction of π^* to U, i.e. $\pi^\circledast = Ranking(U \mapsto \pi^*)$.

Lemma 8.15. *All vertices in the unambiguous set U satisfy $|\sigma^2(v) - \pi^\circledast(v)| = O(\epsilon n)$.*

Proof. Since π^* is a ranking the number of vertices $|B|$ between $\pi^*(v)$ and $\sigma^2(v)$ in π^* is at least $|\pi^*(v) - \sigma^2(v)| - 1$. Therefore by Lemma 8.9 we have

$$\begin{aligned}\frac{|\pi^*(v) - \sigma^2(v)| - 1}{(n-1)3^{k-1}}\binom{n-1}{k-1} &\leq b(\pi^*, v, \sigma^2(v)) + b(\pi^*, v, \pi^*(v)) && \text{(Lemma 8.9)} \\ &\leq 2b(\pi^*, v, \sigma^2(v)) && \text{(Optimality of } \pi^*\text{)}.\end{aligned}$$

We proceed

$$\begin{aligned} b(\pi^*, v, \sigma^2(v)) &\le b(\sigma^{\square}, v, \sigma^2(v)) + O(\epsilon n^{k-1}) && \text{(Lemma 8.12, part one)} \\ &\le b(\sigma^1, v, \sigma^2(v)) + O(\epsilon n^{k-1}) + O(\epsilon n^{k-1}) && (8.4) \\ &= O(\epsilon n^{k-1}) && \text{(Definition of } U) \end{aligned}$$

hence we conclude $|\pi^*(v) - \sigma^2(v)| = O(\epsilon n)$.

Finally we conclude

$$\begin{aligned} |\pi^{\circledast}(v) - \sigma^2(v)| &\le |\pi^{\circledast}(v) - \pi^*(v)| + |\pi^*(v) - \sigma^2(v)| = |\pi^{\circledast}(v) - \pi^*(v)| + O(\epsilon n) \\ &\le \frac{k \cdot OPT}{\epsilon \binom{k}{2}\binom{n-1}{k-1}} + O(\epsilon n) && \text{(Lemma 8.14)} \\ &= O(\epsilon n). \end{aligned}$$

□

8.7 Analysis of π^3

Note that all orderings and costs in this section are over U, not V. We note by Lemma 8.14 that $|U| = n - o(n)$.

Lemma 8.16. $\frac{1}{3^{k-1}}(1-2/10)\binom{|U|-2}{k-2} \le \bar{w}^{\sigma^2}_{uv} + \bar{w}^{\sigma^2}_{vu} \le 2\binom{|U|-2}{k-2}$, *i.e.* $\bar{w}^{\sigma^2}$ *is a weighted FAST instance.*

Proof. We prove the more interesting lower-bound and leave the straightforward proof of the upper bound to the reader. Fix $u, v \in U$. We consider two cases.

If there are at least $|U|/3$ vertices between u and v in σ^2 then we note that by weak fragility every constraint $S \supseteq \{u, v\}$ with all vertices in S between u and v in σ^2 contributes at least 1 to $w_{uv} + w_{vu}$. Therefore $w_{uv} + w_{vu} \ge \binom{|U|/3}{k-2} \ge \frac{1}{2 \cdot 3^{k-2}}\binom{n-2}{k-2}$ for sufficiently large n and small ϵ.

If there are at most $|U|/3$ vertices between u and v in σ^2 then consider constraints with all their vertices either all before or all after u and v. We note that by weak fragility each such constraint $S \supseteq \{u, v\}$ contributes at least 1 to $w_{uv} + w_{vu}$. There are clearly either at least $|U|/3$ vertices before or at least $|U|/3$ vertices after, hence at least $\binom{|U|/3}{k-2} \ge \frac{1}{2 \cdot 3^{k-2}}\binom{n-2}{k-2}$ constraints for sufficiently large n and small ϵ.

We conclude that $w_{uv} + w_{vu} \ge \frac{1}{2 \cdot 3^{k-2}}\binom{n-2}{k-2} \ge \frac{1}{3^{k-1}}\binom{n-2}{k-2}$. The Lemma follows from the definition of $\bar{w}$. □

We define the shorthand $OPT_U = C(\pi^{\circledast})$.

Lemma 8.17. *Assume ranking π and ordering σ satisfy $|\pi(u) - \sigma(u)| = O(\epsilon n)$ for all u. For any u, v, let N_{uv} denote the number of $S \supset \{u, v\}$ such that not all pairs $\{s, t\} \neq \{u, v\}$ are in the same order in σ and π. We have $N_{uv} = O(\epsilon n^{k-2})$.*

Proof. Such a pair $\{s, t\}$ must satisfy $|\pi(s) - \pi(t)| = 2 \cdot O(\epsilon n)$, but few constraints contain such a pair. □

Lemma 8.18. *The following inequalities hold:*

1. $w_{uv}^{\sigma^2} \leq w_{uv}^{\pi^{\circledast}} + O(\epsilon n^{k-2})$
2. $\bar{w}_{uv}^{\sigma^2} \leq (1 + O(\epsilon)) w_{uv}^{\pi^{\circledast}}$

Proof. The only constraints $S \supset \{u, v\}$ that contribute differently to the left- and right-hand sides of the first part are those containing a $\{s, t\} \neq \{u, v\}$ that are a $\sigma^2/\pi^{\circledast}$-inversion. By Lemmas 8.15 and 8.17 we can bound the number of such constraints by $O(\epsilon n^k)$, completing the proof of the first part.

If $w_{uv}^{\pi^{\circledast}} \geq \frac{1}{2 \cdot 3^{k-1}} \binom{|U|-2}{k-2}$ the second part follows from the first part and the trivial fact $\bar{w} \leq w$. Otherwise by the first part we have $w_{uv}^{\sigma^2} < 0.6 \frac{1}{3^{k-1}} \binom{|U|-2}{k-2}$. Therefore by Lemma 8.16 $w_{vu}^{\sigma^2} > 0.2 \frac{1}{3^{k-1}} \binom{|U|-2}{k-2}$ hence $\bar{w}_{uv}^{\sigma^2} = w_{uv}^{\sigma^2} - \min(0.1 \frac{1}{3^{k-1}} \binom{|U|-2}{k-2}, w_{uv}^{\sigma^2}) = \min(w_{uv}^{\sigma^2} - 0.1 \frac{1}{3^{k-1}} \binom{|U|-2}{k-2}, 0) \leq \min(w_{uv}^{\pi^{\circledast}}, 0) \leq w_{uv}^{\pi^{\circledast}}$ using the first part of the Lemma in the penultimate inequality. □

Lemma 8.19.

1. $\bar{C}^{\sigma^2}(\pi^{\circledast}) \leq (1 + O(\epsilon)) \binom{k}{2} OPT_U$
2. $\bar{C}^{\sigma^2}(\pi^3) \leq (1 + O(\epsilon)) \binom{k}{2} OPT_U$
3. $\bar{C}^{\sigma^2}(\pi^3) - \bar{C}^{\sigma^2}(\pi^{\circledast}) = O(\epsilon OPT_U)$

Proof. From the second part of Lemma 8.18 and Lemma 8.7 we conclude that

$$\bar{C}^{\sigma^2}(\pi^{\circledast}) \leq (1 + O(\epsilon)) C^{\pi^{\circledast}}(\pi^{\circledast}) = (1 + O(\epsilon)) \binom{k}{2} OPT_U.$$

proving the first part of this Lemma.

The PTAS for FAST guarantees

$$\bar{C}^{\sigma^2}(\pi^3) \leq (1 + O(\epsilon)) \bar{C}^{\sigma^2}(\pi^{\circledast}), \tag{8.5}$$

which combined with the first part of this Lemma yields the second part.

Finally the first part of Lemma 8.18 followed by the first part of this Lemma imply

$$\bar{C}^{\sigma^2}(\pi^3) - \bar{C}^{\sigma^2}(\pi^{\circledast}) \leq O(\epsilon) C^{\sigma^2}(\pi^{\circledast}) \leq O(\epsilon OPT_U),$$

completing the proof of the third part of this Lemma. □

Lemma 8.20. $d(\pi^3, \pi^{\circledast}) = O(OPT_U / n^{k-2})$

Proof. π^3 and $\pi^{\circledast}$ both have cost at most $2OPT_U$ (Lemma 8.19, first and second parts) for the FAST instance $\bar{w}^{\sigma^2}$ (Lemma 8.16). □

Lemma 8.21. *We have* $|\pi^3(v) - \pi^{\circledast}(v)| = O(\epsilon n)$ *for all* $v \in U$.

Proof. Fix $v \in U$. In this proof we write w (resp. $\bar{w}$) as a short-hand for w^{σ^2} (resp. $\bar{w}^{\sigma^2}$). Observe that there are at least $(|\pi^3(v) - \pi^{\circledast}(v)| - 1)$ vertices between $\pi^3(v)$ and $\pi^{\circledast}(v) + 1/2$ in π^3. Any such vertex u must contribute w_{uv} to one of $b^{\bar{w}}(\pi^3, v, \pi^{\circledast}(v) + 1/2)$ and $b^{\bar{w}}(\pi^3, v, \pi^3(v))$ and contribute w_{vu} to the other. By Lemma 8.16 and local optimality of π^3 we have

$$\begin{aligned}(|\pi^3(v) - \pi^{\circledast}(v)| - 1)\frac{(1 - 2/10)}{3^{k-1}}\binom{|U| - 2}{k - 2} &\leq b^{\bar{w}}(\pi^3, v, \pi^{\circledast}(v) + 1/2) + b^{\bar{w}}(\pi^3, v, \pi^3(v)) \\ &\leq 2b^{\bar{w}}(\pi^3, v, \pi^{\circledast}(v) + 1/2).\end{aligned}$$

Now apply Corollary 8.13

$$b^{\bar{w}}(\pi^3, v, \pi^{\circledast}(v) + 1/2) \leq b^{\bar{w}}(\pi^{\circledast}, v, \pi^{\circledast}(v)) + 2\sqrt{d(\pi^{\circledast}, \pi^3)}2\binom{|U| - 2}{k - 2}$$

and then recall $\sqrt{d(\pi^{\circledast}, \pi^3)} = O(\epsilon n)$ by Lemma 8.20 and the assumption that OPT is small.

Next

$$\begin{aligned}b^{\bar{w}}(\pi^{\circledast}, v, \pi^{\circledast}(v)) &\leq (1 + O(\epsilon))b^{w^{\pi^{\circledast}}}(\pi^{\circledast}, v, \pi^{\circledast}(v)) && \text{(Second part of Lemma 8.18)} \\ &= (1 + O(\epsilon))b(\pi^{\circledast}, v, \pi^{\circledast}(v)) && \text{(Lemma 8.7)} \qquad (8.6)\end{aligned}$$

Finally

$$\begin{aligned}b(\pi^{\circledast}, v, \pi^{\circledast}(v)) &\leq b(\sigma^1, v, \sigma^2(v)) + O(n^{k-2}(\epsilon n + \sqrt{\epsilon^2 n^2})) && \text{(Lemmas 8.12, 8.11 and 8.15)} \\ &= O(\epsilon n^{k-1}) && (v \in U).\end{aligned}$$

which completes the proof of the Lemma. □

Lemma 8.22. $C(\pi^3) \leq (1 + O(\epsilon))OPT_U$.

Proof. First we claim that

$$|(C(\pi^3) - C(\pi^{\circledast})) - (C^{\sigma^2}(\pi^3) - C^{\sigma^2}(\pi^{\circledast}))| \leq E_1, \qquad (8.7)$$

where E_1 is the number of constraints that contain one pair of vertices u, v in different order in π^3 and $\pi^{\circledast}$ and another pair $\{s, t\} \neq \{u, v\}$ with relative order in π^3, $\pi^{\circledast}$ and σ^2 not all equal. Indeed constraints ordered identically in π^3 and $\pi^{\circledast}$ contribute zero to both sides of (8.7), regardless of σ^2. Consider some constraint S containing a $\pi^3(v)/\pi^{\circledast}$-inversion $\{u, v\} \subset S$. If the restrictions of the three orderings to S are identical except possibly for swapping u, v then S contributes equally to both sides of (8.7), proving the claim.

To bound E_1 observe that the number of inversions u, v is $d(\pi^3, \pi^{\circledast}) \equiv D$. For any u, v Lemmas 8.21, 8.15 and 8.17 allow us to show at most $O(\epsilon n^{k-2})$ constraints contribute, so $E_1 = O(D\epsilon n^{k-2}) = O(\epsilon OPT_U)$ (Lemma 8.20).

Finally bound $C^{\sigma^2}(\pi^3) - C^{\sigma^2}(\pi^{\circledast}) = \bar{C}^{\sigma^2}(\pi^3) - \bar{C}^{\sigma^2}(\pi^{\circledast}) = O(\epsilon OPT_U)$, where the equality follows from the definition of w and the inequality is the third part of Lemma 8.19. □

8.8 Analysis of π^4

We now prove Theorem 8.4, that is

$$C(\pi^4) \leq (1 + O(\epsilon))OPT. \tag{8.8}$$

We consider three contributions to these costs separately: constraints with 0, 1, or 2+ vertices in $V \setminus U$.

The contribution of constraints with 0 vertices in $V \setminus U$ to the left- and right-hand sides of (8.8) are clearly $C(\pi^3)$ and $C(\pi^{\circledast})$ respectively. We showed $C(\pi^3) \leq C(\pi^{\circledast}) + O(\epsilon)OPT_U$ in Lemma 8.22.

Second we consider the contribution of constraints with exactly 1 vertex in $V \setminus U$. Consider some $v \in V \setminus U$. We want to compare $b(\pi^3, v, \sigma^4(v))$ and $b((U \mapsto \pi^*), v, \pi^*(v))$. Let p be the half-integer so that $Ranking(v \mapsto p | U \mapsto \pi^{\circledast}) = Ranking(v \mapsto \pi^*(v) | U \mapsto \pi^*)$. The algorithm's greedy choice minimizes $b(\pi^3, v, \sigma^4(v))$ so $b(\pi^3, v, \sigma^4(v)) \leq b(\pi^3, v, p)$. Now using Lemmas 8.12 and 8.20 we have $b(\pi^3, v, p) \leq b(\pi^{\circledast}, v, p) + O(\sqrt{d(\pi^3, \pi^{\circledast})}n^{k-2}) = b(\pi^{\circledast}, v, p) + O(\sqrt{OPT/n^k}n^{k-1})$. Note $b(\pi^{\circledast}, v, p) = b((U \mapsto \pi^*), v, \pi^*(v))$. Let $\gamma = OPT/n^k$. We conclude by Lemma 8.14 that the contribution of constraints with exactly 1 vertex in $V \setminus U$ is $O(|V \setminus U|\sqrt{OPT/n^k}n^{k-1}) = O(\frac{\gamma^{3/2}n^k}{\epsilon}) = O(\epsilon OPT)$.

Finally by Lemma 8.14 there are at most $|V \setminus U|^2 n^{k-2} = O((\frac{\gamma}{\epsilon})^2 n^2 n^{k-2}) = O(\epsilon^2 OPT)$ constraints containing two or more vertices from $V \setminus U$.

This ends the analysis of our algorithm.

Chapter 9

Ranking MIN-CSPs: exact algorithms

9.1 Introduction

The results presented in this chapter are joint work with Marek Karpinski. They previously appreared in Karpinski and Schudy [2009a].

In this chapter we give exact algorithms for many of the ranking problems considered in Chapters 7 and 8.

Theorem 9.1. *There exists a parameterized subexponential algorithm for* FAST *with runtime* $2^{O(\sqrt{K})} + n^{O(1)}$ *for* $OPT \leq K$*. A variant of the algorithm uses* $2^{O(\sqrt{K}\log K)} + n^{O(1)}$ *time and* $n^{O(1)}$ *space.*

Both results in Theorem 9.1 improve the best up to now known parameterized runtime bound of Alon, Lokshtanov and Saurabh Alon et al. [2009] for the feedback arc set tournament problem by a $\Theta(\log K)$ factor in the exponent. Feige [2009] indepently discovered Theorem 9.1, using a different algorithm. We also give improved results for the closely related problem of *Kemeny rank aggregation (KRA)*; see e.g. Ailon [2007], Mathieu and Schudy [2009].

Theorem 9.2. *Let* m *be the number of input rankings (voters),* n *the number of candidates, and* $OPT \leq m\binom{n}{2}$ *the (unscaled) optimum value. There exists a parameterized subexponential algorithm for Kemeny Rank Aggregation with runtime and space* $2^{O(\sqrt{K})} + n^{O(1)}$ *for* $OPT/m \leq K$*. A variant uses* $2^{O(\sqrt{K}\log(K))} + n^{O(1)}$ *time and* $n^{O(1)}$ *space.*

Note that our bound in Theorem 9.2 is based on an upper-bound K on the scaled optimum value OPT/m, that is the average distance from input rankings to the output ranking. This is arguably a more natural parameter than OPT itself. The best previously known runtime was $n^{O(1)} + 2^{O(K)}$

Betzler et al. [2009].[1]

We also give the first fixed-parameter tractability result for our fragile ranking generalization for arity 3.

Theorem 9.3. *There exist parameterized subexponential algorithms for all fragile rank CSPs on tournaments with arity three (e.g. 3-*FAST *and* BETWEENNESSTOUR*) with runtime* $2^{O(\sqrt{K/n})} \cdot n^{O(1)}$ *for* $OPT \leq K$*.*

For betweenness the previously best known runtime was $2^{O(K^{1/3} \log K)}$ Saurabh [2009]. Our result is better by a log factor in the exponent for the largest possible $K = \Theta(n^3)$ and even better for smaller K. Interestingly we can solve instances with K as large as $\Theta(n \log^2 n)$ in polynomial time!

9.2 Algorithms and analysis

Our exact algorithms are based on a few additional ideas. We describe our techniques for exact FAST here and defer discussion of the other problems until later. Firstly any two low-cost rankings for a FAST problem are nearby in Kendall-Tau distance. Secondly two rankings that are Kendall-Tau distance D apart are equivalent to within additive $O(\sqrt{D})$ in how good each location for each a vertex is (Corollary 8.13). Thirdly a consequence of fragility is that most vertices (in a low-cost instance) have a vee-shaped cost versus position curve (Lemma 9.5), and optimal rankings are locally optimal so we know that each vertex belongs at the bottom of its curve. The uncertainty in this curve by $\sqrt{D}$ causes an uncertainty in the optimal position also around $\sqrt{D}$ (Lemma 9.4). Our algorithm simply computes uncertainties $r(v)$ in the positions of all of the vertices v and solves a dynamic program for the optimal ranking that is near a particular constant-factor approximate ranking. We remark that Braverman and Mossel Braverman and Mossel [2008] and Betzler et al. Betzler et al. [2008, 2009] previously applied dynamic programming to FAST and KRA.

The kernelization algorithm of Dom et al. Dom et al. [2006] allows an arbitrary FAST instance of cost $OPT \leq K$ to be reduced to an equivalent one with $O(K^2)$ vertices in time $n^{O(1)}$. There is a kernelization algorithm for KRA in Betzler et al. [2009], but it produces an instance of size $O(OPT)$, not the desired $O(OPT/m)$. To get the desired kernel we generalize Betzler et al. [2009] slightly, creating the kernel by repeatedly discarding Concorcet winners (ranking them first) and Condorcet losers (ranking them last).

Lemma 9.4. *In Algorithm 9.1 we have* $|\pi^*(v) - \pi^4(v)| \leq r(v)$ *for all* $v \in V$ *where* π^* *is an optimal ranking of* V*.*

Proof. We have a tournament so $d(\pi^*, \pi^4) \leq C(\pi^*) + C(\pi^4) \leq 2C(\pi^4)$. By Corollary 8.13 therefore

$$|b(\pi^*, v, j + 1/2) - b(\pi^4, v, j + 1/2)| \leq 2\sqrt{2C(\pi^4)} \tag{9.1}$$

[1] Stated therein as runtime $2^{O(d_a)}$ where d_a is the average pairwise Kendall-Tau distance between the input rankings. We note that $d_a = \Theta(OPT/m)$ follows easily from the triangle inequality; see e.g. the classic proof that picking a random input ranking is a 2-approximation in expectation.

Input: Vertex set V_0, constraint system c_0.

1: Compute a kernel with vertex set V, $|V| = O(K^2)$, and constraint system c (used for rest of algorithm) Dom et al. [2006], Betzler et al. [2009]. Hereafter interpret notations such as $C(\cdot)$ and n relative to instance V, c, not V_0, c_0.
2: Sort the kernel V,c by wins Coppersmith et al. [2006], yielding ranking π^4 of V.
3: Set $r(v) = 4\sqrt{2C(\pi^4)} + 2b(\pi^4, v, \pi^4(v))$ for all $v \in V$.
4: Use dynamic programming or divide-and-conquer (Details: Lemma 9.6) to find the optimal ranking π^5 with $|\pi^5(v) - \pi^4(v)| \leq r(v)$ for all v.
5: "Undo" the kernel step, extending ranking π^5 of the kernel into a ranking of V_0 as described in Dom et al. [2006], Betzler et al. [2009].

Figure 9.1: Exact algorithm for FAST, where $K = OPT$, and KRA, where $K = OPT/m$. If dynamic programming is used in the last line the runtime and space are both $n^{O(1)} + 2^{O(\sqrt{K})}$. If divide-and-conquer is used the runtime is $n^{O(1)} + 2^{O(\sqrt{K}\log K)}$ and the space is $n^{O(1)}$.

Input: Vertex set V

1: Use Algorithm 8.3 to construct a 2-approximate ranking π^{good}.
2: **for** each π^4 considered by our PTAS when constructing a 2-approximation **do**
3: **if** $C(\pi^4) \leq 2C(\pi^{good})$ **then**
4: Set $r(v) = \alpha_1\sqrt{C(\pi^4)/n} + \alpha_2 b(\pi^4, v, \pi^4(v))/n$ for all $v \in V$, where α_1 and α_2 are absolute constants.
5: Use dynamic programming (see Lemma 9.6) to find the optimal ranking π^5 with $|\pi^5(v) - \pi^4(v)| \leq r(v)$ for all v.
6: **end if**
7: **end for**
8: Return the best of the π^5 rankings.

Figure 9.2: Exact algorithm for weak fragile ranking 3-CSPs in tournaments. The runtime is $n^{O(1)}2^{O(\sqrt{OPT/n})}$.

for any $j \in \mathbb{Z}$.

Fix $v \in V$. We conclude

$$\begin{aligned}
|\pi^*(v) - \pi^4(v)| &\leq b(\pi^4, v, \pi^4(v)) + b(\pi^4, v, \pi^*(v)) && \text{(Fragility)}\\
&= b(\pi^4, v, \pi^*(v) + 1/2) + b(\pi^4, v, \pi^4(v) + 1/2) && (\pi^4 \text{ is a ranking})\\
&\leq b(\pi^*, v, \pi^*(v) + 1/2) + 2\sqrt{2C(\pi^4)} + b(\pi^4, v, \pi^4(v) + 1/2) && \text{(By (9.1))}\\
&\leq b(\pi^*, v, \pi^4(v) + 1/2) + 2\sqrt{2C(\pi^4)} + b(\pi^4, v, \pi^4(v) + 1/2) && \text{(Optimality of } \pi^*)\\
&\leq 4\sqrt{2C(\pi^4)} + 2b(\pi^4, v, \pi^4(v) + 1/2) && \text{(By (9.1))}\\
&= r(v) && \text{(Definition of } r(v)).
\end{aligned}$$

□

Lemma 9.5. *In Algorithm 9.1 we have* $\max_j |\{ v \in V \ : \ |\pi^4(v) - j| \leq r(v) \}| = O(\sqrt{OPT})$.

Proof. Fix j. Let $R = \{ v \in V \ : \ |\pi^4(v) - j| \leq r(v) \}$, the cardinality of which we are trying to bound. We say $v \in V$ is *pricey* if $b(\pi^4, v, \pi^4(v)) > \sqrt{2C(\pi^4)}$. Clearly (see also proof of Lemma 8.8) $2C(\pi^4) = \sum_v b(\pi^4, v, \pi^4(v)) \geq (\text{number pricey})\sqrt{2C(\pi^4)}$ hence the number of pricey vertices is at

most $2C(\pi^4)/(\sqrt{2C(\pi^4)} = \sqrt{2C(\pi^4)}$. All non-pricey vertices in R have $|\pi^4(v) - j| \leq 2 \cdot \sqrt{2C(\pi^4)}$, so at most $2\sqrt{2C(\pi^4)} + 1$ non-pricey vertices are in R. We conclude $|R| \leq 3\sqrt{2C(\pi^4)} + 1 = O(\sqrt{OPT})$ since π^4 is a 5-approximation Coppersmith et al. [2006]. □

Lemma 9.6. *For $k \in \{2, 3\}$ there is a dynamic program that finds the optimal ranking π^5 with $|\pi^5(v) - \pi^4(v)| \leq r(v)$ for all v, with space and runtime $O(|V|^k)2^\psi$ where $\psi = \max_j |\{ v \in V \ : \ |\pi^4(v) - j| \leq r(v) \}|$. A divide and conquer variant has runtime $O(|V|^k)2^{O(\psi \log |V|)}$ and $|V|^{O(1)}$ space.*

Proof. Say that a set $S \subseteq V$ is *valid* if it contains all vertices v with $\pi^4(v) \leq |S| - r(v)$ and no vertex v with $\pi^4(v) > |S| + r(v)$. Observe that for any s the valid sets of size s are uncertain about at most ψ vertices, hence there are at most $n2^\psi$ valid sets.

We say that a ranking π of valid set S is *valid* if $\{ v \ : \ \pi(v) \leq j \}$ is a valid set for all $0 \leq j \leq |S|$. It is easy to see that a ranking π is valid if and only if satisfies $|\pi(v) - \pi^4(v)| \leq r(v)$ for all v.

For any ranking π over S let $C'(\pi)$ denote the portion of the cost shared by all orderings with prefix π. That is, the cost of all constraints with at most 1 vertex outside S.[2] One can easily see the following optimal substructure property: prefixes of an optimal (w.r.t. C') valid ranking are optimal (w.r.t. C') valid rankings themselves.

For any valid set S let $\kappa(S)$ denote the C' cost of the optimal (w.r.t. C') valid ranking of S. The dynamic program for $k = 2$ is

$$\kappa(S) = \min_{v \in S : S \setminus \{v\} \text{ is valid}} \left[C'(S \setminus \{v\}) + \sum_{q \in V \setminus S} c(v \mapsto 1 \,|\, q \mapsto 2) \right].$$

and for $k = 3$

$$\kappa(S) = \min_{v \in S : S \setminus \{v\} \text{ is valid}} \left[C'(S \setminus \{v\}) + \sum_{u \in S \setminus \{v\}} \sum_{q \in V \setminus S} c(u \mapsto 1 \,|\, v \mapsto 2 \,|\, q \mapsto 3) \right].$$

The space-efficient variant evaluates κ using divide and conquer instead of dynamic programming, similar to Dom et al. [2006]. Details deferred. □

Proof of Theorems 9.1 and 9.2. Algorithm 9.1 is correct by Lemma 9.4. Lemmas 9.5 and 9.6 allow us to bound the runtime and space requirements of the dynamic program. □

Lemma 9.7. *During the iteration of Algorithm 9.2 that guesses σ^0 correctly we have $|\pi^*(v) - \pi^4(v)| \leq r(v)$ for all $v \in V$ where π^* is an optimal ranking of V.*

Proof. Let ϵ be the error parameter that has our PTAS giving a 2-approximation. By Lemma 8.20 we have $d(\pi^{\circledast}, \pi^3) = O(OPT/n^{3-2})$. This together with Lemma 8.14 imply that

$$d(\pi^*, \pi^4) = O(OPT/n^{3-2} + n \cdot OPT/(\epsilon n^{3-1})) = O(OPT/(\epsilon n))$$

By Lemma 8.12 therefore

$$|b(\pi^*, v, j + 1/2) - b(\pi^4, v, j + 1/2)| = O(n\sqrt{OPT/(\epsilon n)}) \tag{9.2}$$

[2] For $k = 2$ (FAST) it would be more natural to use $C(\pi)$ instead, but this works better for $k = 3$.

for any $j \in \mathbb{Z}$.

Fix $v \in V$. We conclude

$$
\begin{aligned}
&|\pi^*(v) - \pi^4(v)| \frac{1}{(n-1)3^2}\binom{n-1}{2} \\
&\leq b(\pi^4, v, \pi^4(v) + 1/2) + b(\pi^4, v, \pi^*(v) + 1/2) && \text{(Lemma 8.9)} \\
&\leq b(\pi^*, v, \pi^*(v) + 1/2) + O(\sqrt{nC(\pi^4)/\epsilon}) + b(\pi^4, v, \pi^4(v) + 1/2) && \text{(By (9.2))} \\
&\leq b(\pi^*, v, \pi^4(v) + 1/2) + O(\sqrt{nC(\pi^4)/\epsilon}) + b(\pi^4, v, \pi^4(v) + 1/2) && \text{(Optimality of } \pi^*) \\
&\leq O(\sqrt{nC(\pi^4)/\epsilon}) + 2b(\pi^4, v, \pi^4(v) + 1/2) && \text{(By (9.1))} \\
&= r(v)\frac{1}{(n-1)3^2}\binom{n-1}{2} && \text{(Definition of } r(v)).
\end{aligned}
$$

□

Lemma 9.8. *In Algorithm 9.2 we have* $\max_j |\{ v \in V \ : \ |\pi^4(v) - j| \leq r(v) \}| = O(\sqrt{C(\pi^4)/n})$.

Proof. We proceed analogously to the proof of Lemma 9.5. Fix j. Let $R = \{ v \in V \ : \ |\pi^4(v) - j| \leq r(v) \}$, whose cardinality we are trying to bound. We say $v \in V$ is *pricey* if $b(\pi^4, v, \pi^4(v))/n > \sqrt{2C(\pi^4)/n}$. Clearly (see also proof of Lemma 8.8) $3C(\pi^4) = \sum_v b(\pi^4, v, \pi^4(v)) \geq (\text{number pricey}) n\sqrt{2C(\pi^4)/n}$ hence the number of pricey vertices is at most $3C(\pi^4)/(\sqrt{2nC(\pi^4)}) = \sqrt{2C(\pi^4)/n}$. All non-pricey vertices in R have $|\pi^4(v) - j| \leq 2 \cdot \sqrt{2C(\pi^4)/n}$, so at most $2\sqrt{2C(\pi^4)/n} + 1$ non-pricey vertices are in R. We conclude $|R| \leq 3\sqrt{2C(\pi^4)/n} + 1 = O(\sqrt{C(\pi^4)/n})$. □

Proof of Theorem 9.3. Lemmas 9.6 and 9.8, plus the test of the "if", allow us to bound the runtime and space requirements of the dynamic program used by Algorithm 9.2 by $n^{O(1)}2^{O(\sqrt{C(\pi^{good})/n})}$, which is of the correct order since $C(\pi^{good}) \leq 2C(\pi^*)$. The for loop is over a constant number of options and hence does not impact the runtime.

For correctness we focus on the iteration of Algorithm 9.2 that guesses σ^0 correctly. The approximation guarantee of our PTAS holds for this iteration so we have $C(\pi^4) \leq 2C(\pi^*) \leq 2C(\pi^{good})$ and hence the "if" is passed. By Lemma 9.4 π^* is among the orders the dynamic program considers. □

Chapter 10

Conclusions

We introduced the concept of *fragile* constraints for constraint satisfaction problems and ranking constraint satisfaction problems. We showed how combining our fragility concept with various forms of density assumptions allow us to develop polynomial-time approximation schemes for a variety of problems.

One open problem related to this thesis stands out both in potential for applicability and in apparent difficulty: is there a polynomial-time approximation scheme for partial rank aggregation? *Partial rank aggregation* is a generalization of Kemeny rank aggregation where rankings may give several candidates the same rank, i.e. ties. Partial rank aggregation is especially interesting since voters frequently do not rank the entire universe; for example search engines give only the top results. A 1.5-approximation of partial rank aggregation exists [Ailon, 2007] which reduces to a form of weighted feedback arc set where the weights satisfy the triangle inequality (i.e. directed metric). The additive error algorithms that we use as subroutines (e.g. Theorem 2.2) also generalize to triangle-inequality instances Fernandez de la Vega et al. [2005]. We suspect that there is a PTAS for partial rank aggregation, but the extension of our PTAS for Kemeny rank aggregation to this problem is not obvious.

Another interesting open question involves the noisy correlation clustering model and semi-definite program described in Chapter 5. In that chapter we showed that if the base clusters are *all* of size at least $c \cdot \sqrt{n}$ then the semi-definite program reconstructs the base clustering exactly. The open question is what happens when some base clusters are smaller than the threshold while others are larger. It is fairly easy to see that the optimal SDP solution X^* in that case will not be integral, even within the large clusters. However we conjecture that rounding each entry of X^* to the nearest integer would yield a graph whose large connected components are precisely the large base clusters. Analyzing this is tricky because the effect of the noise on the objective function of $\Theta(n\sqrt{n})$ is much greater than the effect on the objective of putting a single cluster together, which is $\Theta((\sqrt{n})^2) = \Theta(n)$.

Bibliography

N. Ailon. Aggregation of partial rankings, p-ratings and top-m lists. In *Procs. 18th ACM-SIAM SODA*, pages 415–424, 2007. Journal version to appear in *Algorithmica*. 85, 114, 119

N. Ailon and N. Alon. Hardness of fully dense problems. *Inf. Comput.*, 205(8):1117–1129, 2007. 6, 10, 100, 101

N. Ailon and M. Charikar. Fitting tree metrics: Hierarchical clustering and phylogeny. In *Proc. 46th IEEE FOCS*, pages 73–82, 2005. 39, 40

N. Ailon and M. Mohri. An efficient reduction of ranking to classification. In *Procs. 21st COLT*, pages 87–97, 2008. 2, 76, 78, 84, 97

N. Ailon, M. Charikar, and A. Newman. Aggregating inconsistent information: ranking and clustering. *J. ACM*, 55(5):Article 23, 2008. Conference version in STOC '05. 2, 38, 48, 53, 59, 84, 85, 86

F. Alizadeh. Interior point methods in semidefinite programming with applications to combinatorial optimization. *SIAM J. on Optimization*, 5(1):13–51, 1995. 64, 66

N. Alon. Ranking tournaments. *SIAM J. Discrete Math.*, 20(1):137–142, 2006. 2, 84

N. Alon and J. H. Spencer. *The Probabilistic Method*, chapter 7.8, pages 115–116. Wiley, third edition edition, 2008. 57

N. Alon, M. Krivelevich, and B. Sudakov. Finding a large hidden clique in a random graph. In *SODA '98: Procs. 9th ACM-SIAM Symposium on Discrete Algorithms*, pages 594–598, 1998. 47

N. Alon, W. Fernandez de la Vega, R. Kannan, and M. Karpinski. Random Sampling and Approximation of MAX-CSP Problems. In *Proc. 34th ACM STOC*, pages 232–239, 2002. Journal version in *J. Comput. System Sciences*, 67(2):212–243, 2003. 5, 10, 26

N. Alon, D. Lokshtanov, and S. Saurabh. Fast FAST. In *Procs. 36th ICALP, Part I, LNCS*, volume 5555, pages 49–58, 2009. 85, 114

S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and the hardness of approximation problems. *J. ACM*, 45(3):501. 5

S. Arora, D. Karger, and M. Karpinski. Polynomial Time Approximation Schemes for Dense Instances of NP-Hard Problems. In *Proc. 27th ACM STOC*, pages 284–293, 1995. Journal version in *J. Comput. System Sciences*, 58(1):193–210, 1999. 6, 10, 28, 30

S. Arora, A. M. Frieze, and H. Kaplan. A new rounding procedure for the assignment problem with applications to dense graph arrangement problems. *Mathematical Programming*, 92(1):1–36, 2002. 87

V. Arya, N. Garg, R. Khandekar, A. Meyerson, K. Munagala, and V. Pandit. Local search heuristics for k-median and facility location problems. *SIAM J. Comput.*, 33(3):544–562, 2004. 6

F. Baker and L. Hubert. Applications of combinatorial programming to data analysis: seriation using symmetric proximity measures. *British J. Math. Statis. Psych.*, 30:154–164, 1977. 83

N. Bansal, A. Blum, and S. Chawla. Correlation clustering. *Mach. Learn.*, 56(1-3):89–113, 2004. 3, 38, 39, 47, 49

J. Bartholdi, III, C. Tovey, and M. Trick. Voting schemes for which it can be difficult to tell who won the election. *Social Choice and Welfare*, 6:157–165, 1989. 2, 85

C. Bazgan, W. Fernandez de la Vega, and M. Karpinski. Polynomial Time Approximation Schemes for Dense Instances of Minimum Constraint Satisfaction. *Random Structures and Algorithms*, 23 (1):73–91, 2003. 28, 30

A. Ben-Dor, R. Shamir, and Z. Yakhini. Clustering gene expression patterns. *Journal of Computational Biology*, 6(3-4):281–297, 1999. 3, 38, 47

B. Berger and P. Shor. Tight bounds for the maximum acyclic subgraph problem. *J. Algorithms*, 25(1):1–18, 1997. 83

M. Bertolacci and A. Wirth. Are approximation algorithms for consensus clustering worthwhile? In *SDM '07: Procs. 7th SIAM International Conference on Data Mining*, 2007. 77

N. Betzler, M. R. Fellows, J. Guo, R. Niedermeier, and F. A. Rosamond. Fixed-parameter algorithms for Kemeny scores. In *AAIM '08: Procs. 4th int'l conf. on Algorithmic Aspects in Information and Management*, pages 60–71, 2008. ISBN 978-3-540-68865-5. 85, 115

N. Betzler, M. R. Fellows, J. Guo, R. Niedermeier, and F. A. Rosamond. How similarity helps to efficiently compute Kemeny rankings. In *AAMAS '09: 8th International Conference on Autonomous Agents and Multiagent Systems*, pages 657–664, 2009. Journal version in Theoretical Computer Science 410 (2009) pp. 4554–4570. 115, 116

B. Bollobás. *Random Graphs*, chapter 11.1. Cambridge University Press, second edition, 2001. 52

R. Boppana. Eigenvalues and graph bisection: An average-case analysis. In *Procs. 28th Foundations of Computer Science*, pages 280–285, 1987. 47

J. Borda. Mémoire sur les élections au scrutin. *Histoire de l'Académie Royale des Sciences*, 1781. 2, 85

M. Braverman and E. Mossel. Noisy sorting without resampling. In *Procs. 19th ACM-SIAM SODA*, pages 268–276, 2008. 47, 85, 115

M. Cai, X. Deng, and W. Zang. An approximation algorithm for feedback vertex sets in tournaments. *SIAM J. Computing*, 30(6):1993–2007, 2001. 85

J. Carlson and D. Stolarski. The Correct Solution to Berlekamp's Switching Game. *Discrete Mathematics*, 287(1–3):145–150, 2004. 28

P. Charbit, S. Thomasse, and A. Yeo. The minimum feedback arc set problem is NP-hard for tournaments. *Combinatorics, Probability and Computing*, 16:1–4, 2007. 2, 84

M. Charikar, V. Guruswami, and A. Wirth. Clustering with qualitative information. *J. Computer and System Sciences*, 71(3):360 – 383, 2005. 3, 4, 38, 39, 52, 85

M. Charikar, K. Makarychev, and Y. Makarychev. On the advantage over random for maximum acyclic subgraph. In *Procs. 48th IEEE FOCS*, pages 625–633, 2007a. 83

M. Charikar, K. Makarychev, and Y. Makarychev. A divide and conquer algorithm for d-dimensional arrangement. In *Procs. 18th ACM-SIAM SODA*, pages 541–546, 2007b. 85

M. Charikar, V. Guruswami, and R. Manokaran. Every Permutation CSP of Arity 3 is Approximation Restitant. In *24th IEEE CCC*, 2009. 100

B. Chor and M. Sudan. A geometric approach to betweenness. *SIAM J. Discrete Math.*, 11(4): 511–523, 1998. 100, 101

W. Cohen and J. Richman. Learning to match and cluster large high-dimensional data sets for data integration. In *KDD '02: Procs. 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 475–480, 2002. 3

W. W. Cohen, R. E. Schapire, and Y. Singer. Learning to order things. *J. Artificial Intelligence Research*, 10:243–270, 1999. 2, 84

T. Coleman and A. Wirth. Ranking tournaments: local search and a new algorithm. In *Procs. 10th workshop on Algorithm Engineering and Experiments (ALENEX)*, pages 133–141, 2008. 84, 86

M. J. Condorcet. *Essai sur l'application de l'analyse à la probabilité des décisions rendues à la pluralité des voix*. 1785. Reprinted by AMS Bookstore in 1972. 2, 85

V. Conitzer. Computer Slater rankings using similarities among candidates. In *Procs. 21st AAAI*, pages 613–619, 2006. 2, 84

V. Conitzer, A. Davenport, and J. Kalagnanam. Improved bounds for computing Kemeny rankings. In *Proc. 21st AAAI*, pages 620–626, 2006. 2, 85

D. Coppersmith, L. Fleischer, and A. Rudra. Ordering by weighted number of wins gives a good ranking for weighted tournaments. In *Procs. 17th ACM-SIAM SODA*, pages 776–782, 2006. 84, 116, 117

S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41:391–407, 1990. 78

E. Demaine, D. Emanuel, A. Fiat, and N. Immorlica. Correlation clustering in general weighted graphs. *Theor. Comput. Sci.*, 361(2):172–187, 2006. 38, 39

C. Demetrescu and I. Finocchi. Break the "right" cycles and get the "best" drawing. In *Procs. 2nd Int'l workshop on Algorithmic Engineering and Experiments (ALENEX)*, pages 171–182, 2000. 83

C. Demetrescu and I. Finocchi. Breaking cycles for minimizing crossings. *J. Experimental Algorithmics*, 6:Article 2, 2001. 83

P. Diaconis and R. Graham. Spearman's footrule as a measure of disarray. *J. Royal Statistical Society*, 39(2):262–268, 1977. 85, 90

I. Dinur and S. Safra. The importance of being biased. In *Procs. 34th ACM STOC*, pages 33–42, 2002. 83

M. Dom, J. Guo, F. Hüffner, R. Niedermeier, and A. Truß. Fixed-parameter tractability results for feedback set problems in tournaments. In *LNCS*, volume 3998, pages 320–331. Springer, 2006. 85, 115, 116, 117

C. Dwork, R. Kumar, M. Naor, and D. Sivakumar. Rank aggegation methods for the web. In *Procs. 10th WWW*, pages 613–622, 2001. The NP-hardness proof is in the online-only appendix available from `http://www10.org/cdrom/papers/577/`. 2, 83, 84, 85, 86

P. Eades, X. Lin, and W. Smyth. A fast and effective heuristic for the feedback arc set problem. *Information Processing Letters*, 47(6):319–323, 1993. 83

M. Elsner and E. Charniak. You talking to me? a corpus and algorithm for conversation disentanglement. In *Proceedings of ACL-08: HLT*, pages 834–842, Columbus, Ohio, June 2008. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/P/P08/P08-1095`. 38, 81

M. Elsner and W. Schudy. Bounding and comparing methods for correlation clustering beyond ILP. In *NAACL-HLT Workshop on Integer Linear Programming for Natural Language Processing (ILP-NLP 2009)*, pages 19–27, 2009. vii, 47, 75

P. Erdös and J. Moon. On sets of consistent arcs in tournaments. *Canadian Math. Bulliten*, 8(3): 269–271, 1965. 83

G. Even, J. Naor, B. Schieber, and M. Sudan. Approximating minimum feedback sets and multicuts in directed graphs. *Algorithmica*, 20(2):151–174, 1998. 83

G. Even, J. S. Naor, S. Rao, and B. Schieber. Divide-and-conquer approximation algorithms via spreading metrics. *J. ACM*, 47(4):585–616, 2000. 83

U. Feige. Faster fast(feedback arc set in tournaments). *CoRR*, abs/0911.5094, 2009. 114

U. Feige and R. Krauthgamer. Finding and certifying a large hidden clique in a semirandom graph. *Random Struct. Algorithms*, 16(2):195–208, 2000. 47, 48, 53, 64

W. Fernandez de la Vega. MAX-CUT has a Randomized Approximation Scheme in Dense Graphs. *Random Struct. Algorithms*, 8(3):187–198, 1996. 10

W. Fernandez de la Vega. On the maximal cardinality of a consistent set of arcs in a random tournament. *J. Combinatorial Theory, Ser. B*, 35(3):328–332, 1983. 83

W. Fernandez de la Vega, M. Karpinski, R. Kannan, and S. Vempala. Tensor decomposition and approximation schemes for constraint satisfaction problems. In *STOC '05: Procs. 37th ACM Symposium on Theory of Computing*, pages 747–754, 2005. 119

W. Fernandez de la Vega, R. Kannan, and M. Karpinski. Approximation of Global MAX–CSP Problems. Technical Report TR06-124, Electronic Colloquim on Computation Complexity, 2006. 10

J. R. Finkel and C. D. Manning. Enforcing transitivity in coreference resolution. In *Proceedings of ACL-08: HLT, Short Papers*, pages 45–48, Columbus, Ohio, June 2008. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/P/P08/P08-2012`. 3, 38, 76

M. Flood. Exact and heuristic algorithms for the weighted feedback arc set problem: a special case of the skew-symmetric quadratic assignment problem. *Networks*, 20(1):1–23, 1990. 83

A. M. Frieze and R. Kannan. Quick approximation to matrices and applications. *Combinatorica*, 19(2):175–220, 1999. 5, 87, 98

Z. Füredi and J. Komlós. The eigenvalues of random symmetric matrices. *Combinatorica*, 1(3): 233–241, 1981. 6, 62, 64, 67, 68

M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, New York, 1979. 1

A. Gionis, H. Mannila, and P. Tsaparas. Clustering aggregation. *ACM Trans. on Knowledge Discovery from Data*, 1(1):Article 4, 2007. 76, 77

I. Giotis and V. Guruswami. Correlation clustering with a fixed number of clusters. *Theory of Computing*, 2(1):249–266, 2006. 3, 30, 39, 40, 43, 49

A. M. Gleason. A search problem in the N-cube. In *Proceedings of Symposia in Applied Mathematics*, pages 175–178, 1960. 29

A. Goder and V. Filkov. Consensus clustering algorithms: Comparison and refinement. In *ALENEX '08: Procs. 10th Workshop on Algorithm Enginering and Experiments*, pages 109–117. SIAM, 2008. 76, 77

M. X. Goemans and D. P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. ACM*, 42(6):1115–1145, 1995. ISSN 0004-5411. doi: http://doi.acm.org/10.1145/227683.227684. 6

A. Greenwald, Z. Li, and W. Schudy. More efficient internal-regret-minimizing algorithms. In *21st Annual Conference on Learning Theory - COLT 2008*, pages 239–250, 2008. 7, 8

M. Grötschel, M. Jünger, and G. Reinelt. Facets of the linear ordering polytope. *Math Programming*, 33:43–60, 1985a. 83

M. Grötschel, M. Jünger, and G. Reinelt. On the acyclic subgraph polytope. *Math Programming*, 33:28–42, 1985b. 83

R. Hassin and S. Rubinstein. Approximations for the maximum acyclic subgraph problem. *Information Processing Letters*, 51(3):133–140, 1994. 83, 86

C. Helmberg. Semidefinite programming for combinatorial optimization. Technical Report ZR-00-34, Konrad-Zuse-Zentrum für Informationstechnik Berlin, 2000. 78

C. Helmberg. *The ConicBundle Library for Convex Optimization*, 2009. Ver. 0.2i from `http://www-user.tu-chemnitz.de /~helmberg/ConicBundle/`. 78

R. Horn and C. Johnson. *Matrix Analysis*, chapter 4.3. Cambridge University Press, 1985. 66

M. Jerrum. Large cliques elude the metropolis process. *Random Structures & Algorithms*, 3(4): 347–359, 1992. 47

T. Joachims. A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization. In *International Conference on Machine Learning (ICML)*, pages 143–151, 1997. 78

T. Joachims and J. Hopcroft. Error bounds for correlation clustering. In *ICML '05: Procs. 22nd International Conference on Machine Learning*, pages 385–392, 2005. 55

H. A. Jung. On subgraphs without cycles in tournaments. In P. Erdös, A. Rényi, and V. Sós, editors, *Combinatorial theory and its applications II*, pages 675–677. North Holland, 1970. 83

M. Jünger. *Ploy hedral Combinators and the Acyclic Subgraph Problem*. Heldermann, Berlin, 1985. 83

R. Karp. Reducibility among combinatorial problems. In *Procs. Complexity of Computer Computations*, pages 85–103, 1972. 83

M. Karpinski and W. Schudy. Linear time approximation schemes for the Gale-Berlekamp game and related minimization problems. In *STOC '09: Proceedings of the 41st annual ACM symposium on Theory of computing*, pages 313–322, 2009a. vii, 7, 28, 38, 100, 114

M. Karpinski and W. Schudy. Approximation schemes for the betweenness problem in tournaments and related ranking problems. arXiv:0911.2214, 2009b.

J. Kemeny. Mathematics without numbers. *Daedalus*, 88:571–591, 1959. 2, 85

J. Kemeny and J. Snell. *Mathematical models in the social sciences*. Blaisdell, New York, 1962. Reprinted by MIT press, Cambridge, 1972. 2, 85

C. Kenyon-Mathieu and W. Schudy. How to rank with few errors: a PTAS for weighted feedback arc set on tournaments. In *Procs. 39^{th} ACM STOC*, pages 95–103, 2007. vii, 83, 84

J. H. Kim and V. H. Vu. Concentration of multivariate polynomials and its applications. *Combinatorica*, 20(3):417–434, 2000. 57, 58

A. Kolla and M. Tulsiani. Playing random and expanding unique games. Unpublished manuscript, 2008. 47

B. Korte. Approximation algorithms for discrete optimization problems. *Ann. Discrete Math*, 4: 85–120, 1979. 83

L. Kucera. Expected complexity of graph partitioning problems. *Discrete Appl. Math.*, 57(2–3): 193–212, 1995. 47

T. Leighton and S. Rao. Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *J. ACM*, 46(6):787–832, 1999. 83

A. Lempel and I. Cederbaum. Minimum feedback arc and vertex set of a directed graph. *IEEE Trans. Circuit Theory*, 13(4):399–403, 1966. 83

M. Luby, A. Sinclair, and D. Zuckerman. Optimal speedup of Las Vegas algorithms. In *Procs. 2^{nd} Israel Symposium on Theory and Computing Systems*, pages 128–133, 1993. 8

I. Malioutov and R. Barzilay. Minimum cut model for spoken lecture segmentation. In *ACL*. The Association for Computer Linguistics, 2006. 75

C. Mathieu and W. Schudy. Yet Another Algorithm for Dense Max Cut: Go Greedy. In *Proc. 19th ACM-SIAM SODA*, pages 176–182, 2008a. 43

C. Mathieu and W. Schudy. Yet another algorithm for dense max cut: Go greedy. In *Procs. 19th ACM-SIAM SODA*, pages 176–182, 2008b. vii, 7, 10

C. Mathieu and W. Schudy. How to rank with fewer errors – a PTAS for feedback arc set in tournaments. In submission `http://www.cs.brown.edu/~ws/papers/fast_journal.pdf`, 2009. 83, 114

C. Mathieu and W. Schudy. Correlation clustering with noisy input. In *Procs. 21st SODA (to appear)*, 2010. Preprint: `http://www.cs.brown.edu/~ws/papers/cluster.pdf`. vii, 47

C. Mathieu, O. Sankur, and W. Schudy. Online correlation clustering. In *Procs. 27th* STACS *(to appear)*, 2010. 7

A. McCallum and B. Wellner. Conditional models of identity uncertainty with application to noun coreference. In *Proceedings of the 18th Annual Conference on Neural Information Processing Systems (NIPS)*, pages 905–912. MIT Press, 2004. URL `http://www.cs.umass.edu/~mccallum/papers/condid-nips2004.pdf`. 38

W. McLendon, III, B. Hendrickson, S. Plimpton, and L. Rauchwerger. Finding strongly connected components in parallel in particle transport sweeps. In *ACM Symposium on Parallel Algorithms and Architectures*, pages 328–329, 2001. 8

W. McLendon, III, B. Hendrickson, S. J. Plimpton, and L. Rauchwerger. Finding strongly connected components in distributed graphs. *J. Parallel Distrib. Comput.*, 65(8):901–910, 2005. 8

F. McSherry. Spectral partitioning of random graphs. In *FOCS '01: Procs. 42nd IEEE Foundations of Computer Science*, page 529, 2001. 47, 48

M. Meila. Comparing clusterings–an information based distance. *Journal of Multivariate Analysis*, 98(5):873–895, May 2007. doi: http://dx.doi.org/10.1016/j.jmva.2006.11.013. URL `http://dx.doi.org/10.1016/j.jmva.2006.11.013`. 79

A. Newman. The maximum acyclic subgraph problem and degree-3 graphs. In *Procs. APPROX*, pages 147–158, 2001. 83

A. Newman. Cuts and orderings: On semidefinite relaxations for the linear ordering problem. In *Procs. APPROX*, pages 195–206, 2004. 83

A. Newman and S. Vempala. Fences are futile: on relaxations for the linear ordering problem. In *Procs. Integer Programming and Combinatorial Optimization (IPCO)*, pages 333–347, 2001. 83

J. Opatrny. Total ordering problems. *SIAM J. Comput.*, 8(1):111–114, 1979. 100, 101

S. Plimpton, B. Hendrickson, S. Burns, and W. McLendon, III. Parallel algorithms for radiation transport on unstructured grids. In *Supercomputing '00: Proceedings of the 2000 ACM/IEEE conference on Supercomputing (CDROM)*, page 25, 2000. 8

K. Reid. On sets of arcs containing no cycles in tournaments. *Canad. Math Bulletin*, 12:261–264, 1969. 83

K. Reid and E. Parker. Disproof of a conjecture of Erdös and Moser on tournaments. *J. Combin. Theory*, 9:225–238, 1970. 83

R. Roth and K. Viswanathan. On the Hardness of Decoding the Gale-Berlekamp Code. *IEEE Transactions on Information Theory*, 54(3):1050–1060, March 2008. 29

M. Rudelson and R. Vershynin. Sampling from large matrices: An approach through geometric functional analysis. *J. ACM*, 54(4):21, 2007. 10

S. Saurabh. Chromatic coding and universal (hyper-) graph coloring families. In *Parameterized Complexity News*, pages 3–4, June 2009. `http://mrfellows.net/Newsletters/2009June_FPT_News.pdf`. 115

F. Schalekamp and A. van Zuylen. Rank aggregation: Together we're strong. In *ALENEX*, pages 38–51, 2009. 2

W. Schudy. Finding strongly connected components in parallel using $O(log^2 n)$ reachability queries. In *SPAA '08: Proceedings of the twentieth annual symposium on Parallelism in algorithms and architectures*, pages 146–151, 2008. 7, 8

W. Schudy. Optimal restart strategies for tree search. In *NESCAI '10: New England Student Colloquim on Artificial Intelligence*, 2010. 7, 8

S. Seshu and M. B. Reed. *Linear Graphs and Electrical Networks*. Addison-Wesley, Reading, MA, 1961. 83

P. D. Seymour. Packing directed circuits fractionally. *Combinatorica*, 15:281–288, 1995. 83

R. Shamir and D. Tsur. Improved algorithms for the random cluster graph model. *Random Structures and Algorithms*, 31(4):418–449, 2007. 47, 48, 49

P. Slater. Inconsistencies in a schedule of paired comparisons. *Biometrika*, 48:303–312, 1961. 2, 83, 84, 86

J. Spencer. *Ten Lectures on the Probabilistic Method*. SIAM, Regional Conference Series, second edition, 1994. 28

J. Spencer. Optimal ranking of tournaments. *Networks*, 1:135–138, 1971. 83

J. Spencer. Optimal ranking of unrankable tournaments. *Period. Math. Hungar.*, 11(2):131–144, 1980. 83

K. Sugiyama, S. Tagawa, and M. Toda. Methods for visual understanding of hierachical system structures. *IEEE Trans. System Man Cybern.*, 11(2):109–125, 1981. 83

C. Swamy. Correlation clustering: maximizing agreements via semidefinite programming. In *SODA '04: Procs. of the 15th ACM-SIAM Symposium on Discrete Algorithms*, pages 526–527, 2004. 38, 48

A. van Zuylen and D. P. Williamson. Deterministic algorithms for rank aggregation and other ranking and clustering problems. In *Procs. Workshop on Approximation and Online Algorithms*, pages 260–273, 2007. 84, 97

A. van Zuylen, R. Hegde, K. Jain, and D. P. Williamson. Deterministic pivoting algorithms for constrained ranking and clustering problems. In *Procs. ACM-SIAM SODA*, pages 405–414, 2007. 38, 48, 84, 97

V. V. Vazirani. *Approximation Algorithms*. Springer, 2001. 6

L. Wang and D. W. Oard. Context-based message expansion for disentanglement of interleaved text conversations. In *Proceedings of NAACL-09 (to appear)*, 2009. 81

P. Young. Optimal voting rules. *The Journal of Economic Perspectives*, 9(1):51–64, 1995. 2, 85

D. H. Younger. Minimum feedback arc sets for a directed graph. *IEEE Trans. Circuit Theory*, 10: 238–245, 1963. 83, 86

S. Zhong and J. Ghosh. Model-based clustering with soft balancing. In *ICDM '03: Proceedings of the Third IEEE International Conference on Data Mining*, page 459, Washington, DC, USA, 2003. IEEE Computer Society. ISBN 0-7695-1978-4. 79