KUNGL
TEKNISKA
HÖGSKOLAN

Some New Randomized Approximation Algorithms

Gunnar Andersson

Stockholm 2000

Doctoral Dissertation
Royal Institute of Technology
Department of Numerical Analysis and Computer Science

Akademisk avhandling som med tillstånd av Kungl Tekniska Högskolan framlägges till offentlig granskning för avläggande av teknisk doktorsexamen fredagen den 26 maj 2000 kl 10.00 i sal E2, Huvudbyggnaden, Kungl Tekniska Högskolan, Lindstedtsvägen 3, Stockholm.

ISBN 91–7170–562–7

TRITA-NA-0009
ISSN 0348-2952
ISRN KTH/NA/R--00/09--SE

KTH Reprocentral, Stockholm 2000

Abstract

The topic of this thesis is approximation algorithms for optimization versions of **NP**-complete decision problems. No exact algorithms with sub-exponential running times are known for these problems, and therefore approximation algorithms with polynomial running times are studied. An approximation algorithm does not necessarily find the optimal solution, but it leaves a guarantee of how far from the optimum the output solution can be in the worst case. This *performance guarantee* is the measure of quality of an approximation algorithm; it should be as close to 1 as possible.

We present new approximation algorithms for several different maximization problems. All problems are essentially constraint satisfaction problems: An instance consists of a set of constraints on groups of variables. The objective is to satisfy as many of the constraints as possible. Most results on such problems are for binary variables; we give some results for binary variables and some where the domain is $\boldsymbol{Z}_p$. A common feature of all such problems is that they can be approximated within a constant factor by picking a variable assignment uniformly at random. Until recently, this was the best known approximation algorithm for many constraint satisfaction problems. Algorithms based on semidefinite programming were introduced by Goemans and Williamson in 1994, and they revolutionized the field. We continue this line of research and use semidefinite programming combined with randomized rounding schemes to obtain algorithms better than picking a solution at random for several different problems: Max Set Splitting, Max 3-Horn Sat, Max E2-Lin mod p, and Max p-Section. When restricted to dense instances, most such problems become easier to approximate. We devise a polynomial time approximation scheme for the family Max Ek-Function Sat mod p of constraint satisfaction problems for which the domain is $\boldsymbol{Z}_p$. We also prove lower bounds on the approximability of Max k-Horn Sat and Max E2-Lin mod p. A lower bound in this context is a proof that it is impossible to approximate a problem within some given performance guarantee unless $\mathbf{P} = \mathbf{NP}$.

Keywords: computational complexity, NP optimization problems, approximation algorithms, performance guarantee, constraint satisfaction problems, semidefinite programming, randomized rounding, lower bounds, polynomial time approximation schemes.

ISBN 91–7170–562–7 • TRITA-NA-0009 • ISSN 0348-2952 • ISRN KTH/NA/R--00/09--SE

Sammanfattning

Ämnet för den här avhandlingen är approximationsalgoritmer för formuleringar av **NP**-fullständiga beslutsproblem som optimeringsproblem. Inga algoritmer med exekveringstid bättre än exponentiell i indatas storlek är kända för dessa problem, och därför studerar man approximationsalgoritmer med polynomisk körtid. En approximationsalgoritm hittar inte nödvändigtvis den bästa lösningen, men den ger en garanti att den lösning den hittar inte är längre från den bästa lösningen än en viss gräns. Denna *approximationsfaktor* är måttet på hur bra en approximationsalgoritm är; ju närmare 1 den är, desto bättre.

Vi presenterar nya approximationsalgoritmer för olika maximeringsproblem. Alla problem vi studerar är väsentligen villkorsproblem av följande typ: Indata består av en variabelmängd och en uppsättning villkor över dessa variabler. Målet är att hitta en tilldelning till variablerna så att så många villkor som möjligt uppfylls. De flesta kända resultat för villkorsproblem är för binära variabler; vi ger några resultat för binära variabler och några för variabler i $\boldsymbol{Z}_p$. En gemensam egenskap hos alla sådana problem är att de kan approximeras inom en konstant faktor genom att välja variablernas värde slumpvis oberoende av varandra. Ända tills nyligen var detta den bästa kända approximationsalgoritmen för många villkorsproblem. 1994 presenterade Goemans och Williamson algoritmer baserade på semidefinit programmering för några sådana problem, och detta arbete innebar ett stort genombrott. Vi fortsätter i samma riktning och använder semidefinit programmering tillsammans med slumpavrundning, och med dessa tekniker konstruerar vi för problemen Max Set Splitting, Max 3-Horn Sat, Max E2-Lin mod p och Max p-Section algoritmer som är bättre än den algoritm som väljer en slumpvis lösning. Om man inskränker sig till täta instanser blir många villkorsproblem lättare att approximera. Vi ger ett approximationsschema för familjen Max Ek-Function Sat mod p av villkorsproblem över $\boldsymbol{Z}_p$. Vi bevisar också undre gränser för approximerbarheten hos Max k-Horn Sat och Max E2-Lin mod p. Med undre gräns menas här ett bevis för att det inte går att approximera ett problem inom en viss approximationsfaktor om inte $\mathbf{P} = \mathbf{NP}$.

Nyckelord: komplexitetsteori, optimeringsproblem i NP, approximationsalgoritmer, approximationsfaktor, villkorsproblem, semidefinit programmering, slumpavrundning, undre gränser, approximationsscheman.

Acknowledgments

First of all, I would like to thank my supervisor *Viggo Kann*. He has always found time whenever I have been in need of assistance. It has been a pleasure working with him. The same goes for *Johan Håstad*, with whom I have had many helpful discussions. His extreme speed of thought has helped me a lot; he has been pointing out ideas to try and reasons why other ideas are not worth pursuing.

Several of the papers that constitute this thesis are the results of collaboration with others; I thank *Lars Engebretsen* and *Johan Håstad* for working with me on these projects.

The theory group at the department of numerical analysis and computer science has been a nice place to work. For this I also thank the rest of the faculty in the group: *Stefan Arnborg*, *Henrik Eriksson*, *Mikael Goldmann*, *Jens Lagergren*, *Karl Meinke*, *Stefan Nilsson*, *Per Svensson*, and *Rand Waltzman*. A special thanks to Rand who got me interested in doing research in theoretical computer science.

A great thanks to my roommates *Lars Engebretsen* and *Lars Ivansson* for a nice time working together. I also want to thank the other PhD students in the group for a fun time studying, doing research, and sometimes juggling, especially *Lars Arvestad*, *Jonas Holmerin*, *Johannes Keukelaar*, *Mats Näslund*, *Anna Redz*, and *Staffan Ulfberg*.

Last but not least, thanks to those who have read drafts of the thesis or parts thereof and provided valuable suggestions: *Elisabet Andersson*, *Sven-Eric Andersson*, *Lars Engebretsen*, *Johan Håstad*, *Viggo Kann*, *Anna Redz*, and *Staffan Ulfberg*.

Contents

Chapter 1

Introduction

1.1 Background

With the invention of computers in the 1940s, mathematicians were faced with a problem only considered occasionally before: How to organize computations to make them run as fast as possible. Before that, only some folklore methods were known, e.g. for multiplying two numbers. There was no real need to study this topic since humans could only perform small calculations by hand. Soon it became clear that the task of finding the best computational methods, *algorithms*, was hard for many problems of interest.

1.2 An example of a computational problem

A company has capacity problems with its web server and has decided to invest in another, identical, web server. The company wants to maximize returns on investment, and the question is now how to use the new server in the best way possible. The plan is to split the files between the two servers.

Having two web servers instead of one immediately doubles the capacity, and it appears as if this should remedy all problems for a while. This is however not necessarily true: The limiting factor is not the capacity but rather congestion; a period with very few requests for web pages can be followed by a burst of requests during a short period of time.

The best use of the second web server is therefore to take care of the peak loads. This can be done by analyzing the access logs for patterns. It turns out that some files are almost always accessed at the same time; e.g. if the web page `index.html` contains the picture `picture.jpg`, then these two files will almost always be requested simultaneously. There are many such correlations due to pictures, frames,

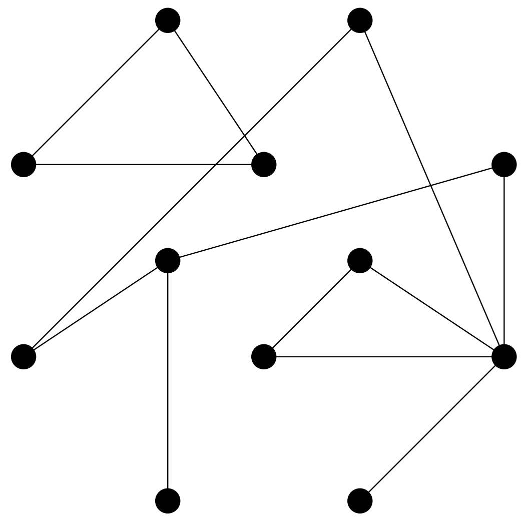

Figure 1.1. Example of an access pattern graph.

style sheets, Java applets, and sound files. This observation can be exploited as follows: Suppose that two files are always accessed at the same time. Then throughput is maximized if they are put on different web servers.

The above ideas can be formalized into the following abstract mathematical problem: Let N be the number of files. For each file, there is a set of files that are often accessed at the same time. This situation can be represented as the network in Figure 1.1. Each circle, or *vertex*, corresponds to a file, and each line, or *edge*, between two vertices means that the corresponding files are often accessed at the same time.

This mathematical structure is called a *graph*, and the computational problem is now how to divide the vertices into two halves of equal size such that the number of edges connecting vertices in different halves is maximized. Such edges are said to be *cut*. A solution to the problem is a partition of the vertices (files) into two halves of equal size. The files in the first half should be placed on the first web server, and the second half on the second web server. This way the number of conflicting requests is minimized.

The model does not take into account that the files are of different sizes. Note that for small files on modern hard disks, the disk seek time (which is constant) exceeds the time to read the file (which depends on its size). As many of the files downloaded from web servers are small, the simple model above might be useful also in an application. The degree to which pairs of files are accessed simultaneously also varies, something which also can be incorporated into the model.

Given that a web server can house thousands of files, a typical access pattern graph would be huge compared to the tiny example above. Finding the best way to split the files between the two web servers is clearly a task for a computer program. But this task turns out to be too hard even for a computer — there are no known algorithms capable of solving this problem in reasonable time when there are thousands of vertices. The example above can be solved by considering all the $\binom{12}{6}/2 = 462$ ways of forming the two halves, but when the number of files is increased to 50, still a very small instance of the problem, the number of combinations grows to a massive 63205303218876. Clearly trying all $\binom{N}{N/2}/2$ possibilities is infeasible once N becomes large. The fact that the number of combinations is large does not necessarily mean that the problem is hard, but for this particular problem we do not know of any method substantially better than trying all combinations. The problem seems to be *computationally hard*.

Our hopes of finding the best way to split the files between the two web servers in any realistic scenario appear to be small. The original problem remains, though: We still have to come up with some scheme for dividing the files between the web servers, we just have to accept that finding the best way of doing so is infeasible. We have to relax our demands and instead of searching for the best solution, which takes unreasonably long to find, settle for a good solution, hopefully much easier to find. How can good be formalized in this context? One way is as follows: The algorithm should always find a solution such that the number of cut edges is at least 50% of the number cut by the best solution. It is easy to devise an algorithm with this property which is fast enough also for very large problems (millions of files). On the other hand, 50% does not sound too impressive. Is it possible to reach, say, 75% or 90% of the number cut by the best solution? This is the kind of question we investigate in this thesis.

1.3 NP-completeness

The foundations of modern theoretical computer science are to some extent based on the advances in optimization theory. This field developed rapidly in the late 1940s and 1950s, much because of Dantzig's discovery of the simplex method for linear programs [27]. Several other important combinatorial problems, such as the assignment problem and the maximum flow problem, could be solved using the simplex method.

An important problem which did not yield to the simplex method is the Travelling Salesman Problem (TSP). In this problem, a salesman has a list of cities that he must visit in some order and then return to the city from which he started. The distance between each pair of cities is known, and the objective is to find the shortest tour. Dantzig, Fulkerson and Johnson [28] solved a TSP instance with 49 cities using the simplex method and an idea based on cutting planes. In spite of this partial success, the TSP seemed harder than the other combinatorial optimization problems considered. New methods, especially cutting planes and a technique called *branch and bound*, made it possible to solve many larger instances, but no systematic method with a proven bound on the running time emerged.

In 1965, Edmonds [29] conjectured that the reason why no efficient algorithm had been discovered for the TSP was that the problem is inherently intractable, meaning that there is no algorithm which solves the problem on all instances in reasonable time. He attached no precise meaning to the term "reasonable time", but a common definition since then is that a reasonable running time is one where the number of steps is a polynomial in the size of the input. If the size of the input is n bits, then the execution of the algorithm will consist of at most kn^c steps where k and c are constants. The class of problems admitting such algorithms is denoted **P**.

In the early 1970s, a better understanding of the relations between the computational problems for which no efficient algorithms were known was developed. It turned out to be fruitful to consider decision problems rather than optimization problems when doing so. A decision problem is a problem where the answer is "yes" or "no", e.g. "Is there a TSP tour of length at most 1000 kilometers?". In applications, optimization problems, e.g. "What is the shortest tour that passes through all the cities?", are more common, but decision problems are obviously closely related. In a breakthrough paper, Cook [22] defined the class **NP** (short for Non-deterministic Polynomial time) and showed that the satisfiability problem for logical formulas is **NP**-complete. **NP** is the class of decision problems for which a "yes"-solution can be verified efficiently, i.e., in time polynomial in the size of the instance. The hardest problems in **NP** are the **NP**-complete problems. As an example of a problem in **NP**, consider the decision problem version of TSP. Suppose that somebody or something, typically a computer program, finds a tour of length at most 1000 kilometers. For a large instance, this might require a long computation even for a fast computer. Having found the solution, it is very easy to convince the user that there really is a solution of length at most 1000 kilometers — if the computer outputs the solution, even a mere human can check by hand that it really visits all cities and that it is not too long. The key property here is that it is easy (polynomial time) to verify that the answer to the question "Is there a TSP tour of length at most 1000 kilometers?" is indeed "yes"; finding the answer in the first place may be much harder. Also note that if the answer is "no, there is no such tour", then it is not apparent how one could verify this without solving the problem from scratch. If the answer is "yes", then the tour itself can be used to check this claim, it can serve as a *certificate* of the claim, but if the answer is "no", no such natural certificate seems to exist. This asymmetry is a key property of **NP**.

The **NP**-complete problems are the hardest in **NP**. Let A be an **NP**-complete problem and B be an arbitrary problem in **NP**. Then for each instance I_B of B, there exists a polynomial-time reduction to an instance I_A of A such that I_A is a "yes"-solution for A if and only if I_B is a "yes"-solution for B. A consequence of this is that if A is **NP**-complete, and there exists a polynomial-time algorithm for A, then all problems in **NP** can be solved in polynomial time. Shortly after

Cook's definition of **NP** and **NP**-completeness, Karp [62] showed that a number of problems, including the TSP, are **NP**-complete. Since then, hundreds of problems have been shown to be **NP**-complete (see [38] for an extensive list). Many of the **NP**-complete problems have been subject to intense study for a long time, without any efficient algorithm being found for any of them. Because of this, it is widely believed that $\mathbf{P} \neq \mathbf{NP}$. Many results in computer science, including this thesis, are based on this assumption.

NP is a set of problems sharing a common property, namely the property that a "yes"-solution can be verified in polynomial time. We say that **NP** is a *complexity class*. By categorizing problems into complexity classes, relationships between problems can often become clearer. There is a plethora of complexity classes based on different categorizations of problems, some based on the time needed to solve a problem and some based on other criteria.

One direction in research in theoretical computer science has always been to search for better algorithms. An algorithm, and the analysis of it, gives an upper bound on the computational resources required to solve a problem. There are several different kinds of computational resources that can be considered. The most common are running time and memory, but in some models resources such as the amount of communication between different subsystems is the natural measure. Another direction is to search for lower bounds. As opposed to improving algorithms, this is a non-constructive effort: A lower bound is a mathematical proof that a problem cannot be solved with less resources than some function of the size of the instance. Proving lower bounds is usually harder than finding algorithms, and for most problems no good lower bounds are known — the best lower bounds are often trivial, stating that an algorithm must read the entire input before producing a result. The ideal situation would be to prove a lower bound matching the best algorithm, or upper bound, for a problem. One example of a problem for which this is the case is the problem of sorting a sequence of n elements from some total order, such as the integers, using only comparisons. For this problem it has been proven that roughly $n \log n$ comparisons are needed, and this is attained by several well-known sorting algorithms (heap sort, merge sort and some versions of quick sort). For most other problems of interest, such as the **NP**-complete problems, not much is known. It has been conjectured that the **NP**-complete problems require time exponential in the size of the input, but proving this seems to be very difficult. The current state of affairs is that it is not even known if the **NP**-complete problems can be solved in time linear in the size of the instance.

1.4 Approximation algorithms

An **NP**-completeness proof for a decision problem is a strong indication that the corresponding optimization problem is hard to solve optimally. One way to attack such problems is to use heuristics, which typically are implementations of various rules-of-thumb combined with searches among some of the possible solutions.

Johnson [54] was the first to introduce approximation algorithms with polynomial running time guaranteeing that the solution output is within some factor of the optimal solution. (This is usually not the case for heuristics.) Since then, approximation algorithms have been derived for a large number of problems.

In Section 1.2 we considered the abstract problem arising from an application in web servers, and mentioned that it is easy to find a division of the files with an objective function value at least 50% of that of the optimal solution. This means that the performance guarantee is 0.5; another way of saying this is that the algorithm is a 0.5-approximation algorithm. A better approximation algorithm has a higher performance guarantee, an algorithm always solving the problem optimally would be a 1-approximation algorithm. The best known approximation algorithm for the web server problem, usually referred to as the Max Bisection problem, is due to Ye [85] and has a performance guarantee of 0.699. For minimization problems, one usually considers the performance ratio, which is the ratio between the value of the solution found and that of the minimal solution. This ratio is at least 1 with equality if and only if the algorithm always finds the optimal solution.

All the **NP**-complete problems are in some sense equally hard to solve, and this also carries over to the optimization versions of these problems — the best algorithms for all such problems have running times that are exponential in the size of the input. The TSP can easily be solved in time $n^2 2^n$ where n is the number of cities. It seems natural to expect that these optimization problems are equally hard also when searching for approximate solutions. This turns out not to be the case; among the **NP** optimization problems there is a vast difference in the degrees of approximability. The reason for this is that the reductions that connect the **NP**-complete problems are only valid for decision problems, they do not preserve the objective function values when applied to the corresponding optimization problems. As an example of this, consider the problems mentioned so far in this chapter: TSP and Max Bisection. It is easy to prove that TSP cannot be approximated within any constant factor unless $\mathbf{P} = \mathbf{NP}$, whereas there exists an algorithm approximating Max Bisection within 0.699.

The **NP**-complete problems arise in a number of different fields such as logics, geometric algorithms, packing etc. This diversity is reflected in the design of approximation algorithms for these problems — many different techniques are used. Standard computer science paradigms such as dynamic programming are common, and combinatorial optimization methods for continuous problems, such as linear programs, are also used frequently.

Lower bounds have been studied also in the context of approximation algorithms. The focus in this field is to prove bounds on how well a problem can be approximated. All such lower bounds are subject to some assumption on the relation between complexity classes, the most common (and in some sense weakest) being that $\mathbf{P} \neq \mathbf{NP}$. Such an assumption is quite natural, since if $\mathbf{P} = \mathbf{NP}$ all optimization problems corresponding to **NP**-complete decision problems could be solved to optimality in polynomial time.

1.5 Recent developments in the field of approximation algorithms

In this section, some of the most important recent results are surveyed, with a bias towards the areas covered in this thesis. The results fall into three distinct categories, and within each category the results are given in the chronological order in which they appeared. Some of the results were first presented at major conferences, and then a few years later published in journals. In such cases, references are made to both versions. For a more thorough account of what is known for different problems, the reader is referred to the book by Ausiello et al. [18] which contains the state of the art (as of 1999) for more than 200 different optimization problems.

1.5.1 Semidefinite relaxations

Many **NP** optimization problems can be cast as integer programs. While this does not immediately lead to good algorithms — solving integer programs is **NP**-hard — it can sometimes be useful in the design of approximation algorithms. Relaxations of integer programs into linear programs have been used in approximation algorithms for a long time. An important property of such relaxations is that they can be solved in polynomial time [58, 64]. For some problems, the natural integer program formulations are non-linear. This complicates matters as solving non-linear optimization problems is hard. There are, however, some classes of such problems that can be solved in polynomial time.

In a breakthrough result, Goemans and Williamson [39, 40] showed how relaxations of quadratic integer programs to semidefinite programs could be used when designing approximation algorithms. Semidefinite programs are a special case of convex programs, and they can be solved (almost) to optimality in polynomial time using interior-point methods (see e.g. [2]). Goemans and Williamson showed how randomized rounding could be applied to the optimal solution to the semidefinite relaxation of the integer quadratic program corresponding to the Max Cut problem, and thereby achieved a 0.878-approximation algorithm. This was a huge improvement from the previous best approximation algorithm, which had a performance guarantee of 0.5 [76]. They applied the same technique to several other problems, obtaining a 0.878-approximation algorithm for Max 2-Sat and a 0.796-approximation algorithm for Max Dicut; the previous best algorithms had performance guarantees of 0.75 and 0.25 respectively.

Several other authors extended Goemans and Williamson's ideas to other problems, and many improved approximation algorithms followed, among others the following: A 0.931-approximation algorithm for Max 2-Sat by Feige and Goemans [31], an algorithm coloring 3-colorable graphs using $O(n^{1/4} \log n)$ colors by Karger, Motwani and Sudan [56, 57], a $\big(1 - 1/k + \Theta(\ln k/k^2)\big)$-approximation algorithm for Max k-Cut by Frieze and Jerrum [35, 36], a 0.875-approximation algorithm for Max 3-Sat by Karloff and Zwick [59]. The result by Karloff and Zwick is especially interesting as their algorithm matches the lower bound by Håstad [52]; their approximation algorithm is provably the best possible for the Max 3-Sat problem unless **P** = **NP**.

Most of the problems for which semidefinite relaxations have been useful have been constraint satisfaction problems: Given a set of local constraints over some larger set, satisfy as many of the constraints as possible. There are some results for other types of problems, e.g. the result by Karger, Motwani and Sudan mentioned above, and semidefinite relaxations is potentially useful in many areas.

1.5.2 New polynomial time approximation schemes

The approximation algorithms mentioned above are all for problems for which a constant performance guarantee strictly smaller than 1 is the best one can hope for, unless **P** = **NP** [13]. For some problems there exist approximation algorithms reaching a performance guarantee (or, for minimization problems, performance ratio) arbitrarily close to 1. Such an algorithm is called a *polynomial time approximation scheme* (PTAS). The majority of the PTASs in the literature are for packing and scheduling problems; an important early result in this domain is the PTAS for bin packing by Karmarkar and Karp [61].

In the last few years, PTASs have been developed for other classes of problems. Arora [9, 10] devised PTASs for a number of important geometric problems in $\boldsymbol{R}^2$ with the Euclidean metric: TSP, Steiner tree, k-MST, and k-TSP. These schemes are all based on dynamic programming and a clever subdivision scheme.

Another class of problems where PTASs have recently been developed is constraint satisfaction problems restricted to dense instances. An instance is dense if it contains a large fraction of the full set of possible constraints. Arora, Karger and Karpinski [11, 12] constructed PTASs for many such problems, including dense Max Cut and dense Max k-Sat. There algorithms are based on exhaustive sampling: Because of the denseness of the instances considered, it suffices to sample a small part of the instance to retrieve the information needed for finding an almost optimal solution. In an independent work, Fernandez de la Vega [34] used another sampling method to construct a PTAS for Max Cut. These techniques were refined by Goldreich, Goldwasser and Ron [41, 42] who obtained PTASs for other problems.

1.5.3 Probabilistically checkable proofs

Research in lower bounds on the approximability of various problems had little success until the early 1990s. Before that, only a few lower bounds were known, and there were no general techniques. Somewhat surprisingly, developments in interactive proofs and program checking came to good use when proving lower bounds. An interactive proof is essentially a game between two players, where one of the players, the *verifier*, has limited computational resources and tries to deduce the truth of some statement by making queries to the other player, the *prover*, who is computationally unbounded. The verifier's goal is to decide probabilistically if $x \in L$ where L is some language; the usual definition of *decide* is that the error probability is some constant strictly less than 1/2. The number of queries made by the verifier is at most a polynomial in the size of x. In two breakthrough papers, Lund et al. [68, 69] and Shamir [79, 80] developed an algebraic way to describe interactive proofs and used this to show that **IP** = **PSPACE**. That is, the languages that can be decided by an interactive proof system are exactly those that can be decided in polynomial space.

In 1991, Feige et al. [32, 33] derived a connection between verifying a probabilistically checkable proof and approximating the Max Clique problem. This implied that finding a constant-factor approximation algorithm is almost **NP**-hard. One year later, Arora and Safra [15, 16] and Arora et al. [13, 14] used coding theory to prove that $\mathbf{NP} = \mathbf{PCP}(\log n, 1)$, the PCP theorem. This result states that a claim that $x \in L$, where L is some **NP**-complete language, can be verified probabilistically by reading some proof Π at a constant number of randomly chosen positions. A consequence of the PCP theorem is that there exist constants $\varepsilon_1, \varepsilon_2$ between 0 and 1 such that Max Clique cannot be approximated within n^{ε_1} unless $\mathbf{P} = \mathbf{NP}$, and that Max 3-Sat cannot be approximated within $1 - \varepsilon_2$ unless $\mathbf{P} = \mathbf{NP}$. Some years earlier, Papadimitriou and Yannakakis [73, 74] had defined the class **Max-SNP**, and showed that Max 3-Sat is complete for this class under so called L-reductions. When combined with the lower bound for Max 3-Sat which follows from the PCP theorem, it follows that there for every **Max-SNP**-complete problem P exists a constant $\varepsilon_P > 0$ such that it is **NP**-hard to approximate the optimum within $1 - \varepsilon_P$. In a sequence of papers, improvements were made to the PCP constructions and therefore also to the lower bounds. For the Max Clique problem, Håstad [50, 51] showed that it is **NP**-hard to find a clique within $n^{1/2-\varepsilon}$ of the size of the largest clique and that it is almost **NP**-hard to find a clique within $n^{1-\varepsilon}$ of the largest clique, for all $\varepsilon > 0$. Håstad also showed that approximating Max 3-Sat within $7/8 + \varepsilon$ is **NP**-hard for all $\varepsilon > 0$ [52], a result which, as mentioned above, is tight — Karloff and Zwick [59] have constructed a 7/8-approximation algorithm.

Håstad's PCP constructions have had implications for many other problems, especially when combined with the new gadget construction techniques by Trevisan et al. [81]. In this work, it is shown how optimal combinatorial reductions using local substructures, so called gadgets, can be found for many problems by solving a linear program. By combining optimal or near-optimal gadgets with PCP constructions, lower bounds can be obtained for new problems without giving explicit PCP constructions.

1.6 Thesis topics

The main theme of this thesis is approximation algorithms for **NP** maximization problems. A common feature of all problems considered is that they are based on constraint satisfaction: The goal is to satisfy as many constraints as possible from a given set, with all constraints sharing the same underlying variable set. In most problems, non-negative weights may be associated with the constraints, but for some only unweighted instances are allowed.

All these problems have the following property in common: It is easy to find an assignment to the variables that gives a solution approximating the optimum within some constant factor. Simply picking a solution uniformly at random from the solution space achieves this on average. Consider the Max Cut problem, where the objective is to partition the vertices of an undirected graph into two parts as to maximize the number of cut edges. A randomized algorithm for this problem is as follows: To decide in which part to put a vertex, toss an unbiased coin. Repeat this for all vertices and a partition is formed. Consider an edge in the graph. It is cut if the first vertex is in part 1 and the second in part 2, or vice versa. The probability of this event is 1/2, and by the linearity of expectation, the average number of edges cut by the random partition is half the number of edges in the graph. This means that the simple approximation algorithm described above is a 1/2-approximation algorithm on average. (It can be converted into a deterministic greedy algorithm with the same performance.) Somewhat surprisingly, this was the best known approximation algorithm for Max Cut until 1994, when Goemans and Williamson [39, 40] gave a 0.878-approximation algorithm. Max Cut is not the only problem for which picking a solution at random gave the best known approximation algorithm for a long time. For Max Sat, this was the case until 1992 when Yannakakis [83, 84] presented a 0.75-approximation algorithm. Yannakakis algorithm was based on a reduction to the maximum flow problem while most other recent papers in the field have used semidefinite programming.

The prime issue in this thesis is investigating problems for which no better approximation algorithm than picking a solution at random were known. The problems are formulated as discrete optimization programs, which are relaxed to semidefinite programs. By applying randomized rounding schemes, and in some cases performing extra post-processing steps, we show how to find an approximate solution with performance guarantee strictly greater than that obtained by picking a solution at random. Most approximation algorithms using semidefinite programming have only considered problems where all variables are binary; one of our results is for linear equations where the domain of all variables is $\boldsymbol{Z}_p$. We also give a PTAS for dense instances of this problem, as well as the generalization where arbitrary functions are allowed. For some of the problems, we also obtain improved lower bounds by constructing new gadgets, both by hand and by adapting the gadget construction techniques by Trevisan et al. [81].

1.7 Overview of the thesis

Chapters 2–4 serve as a background to the rest of the thesis. Notation and terminology from theoretical computer science and mathematics is covered in Chapter 2, while Chapters 3 and 4 contain detailed descriptions of semidefinite programming and techniques for proving lower bounds.

Chapter 5 describes an approximation algorithm for the Max Set Splitting problem. The algorithm is based on a reduction to Goemans-Williamson's algorithm for the Max Cut problem combined with a random perturbation. It is based on the journal article [5], co-authored with Lars Engebretsen. My contribution to this article is approximately 50%.

Chapter 6 investigates the approximability properties of the Max k-Horn Sat problem. For $k = 3$, an approximation algorithm based on semidefinite programming is provided, and for $k \in \{2, 3, 4\}$, lower bounds on the approximability are computed using optimal gadget reductions. The material in this chapter has not been published previously.

Chapter 7 deals with linear equations over $\boldsymbol{Z}_p$ where all equations contain precisely, or at most, two variables. Approximation algorithms for different variations of this problem are given, and a constant lower bound is provided through a gadget reduction. This chapter is based on the full version of the conference paper [7] (submitted for journal publication), co-authored with Lars Engebretsen and Johan Håstad. My contribution to the journal submission is about 30%, but not everything in the journal submission has been included in this thesis, so my contribution to this chapter is about 40%.

Chapter 8 contains an approximation algorithm based on semidefinite programming for the Max p-Section problem. It is based on the conference paper [4].

Chapter 9 describes a polynomial-time approximation scheme for dense instances of the Max Ek-Function Sat mod p problem. The main technique used is exhaustive sampling. The chapter is based on the conference paper [6], co-authored with Lars Engebretsen. My contribution to this chapter is about 50%.

Chapter 10 contains a brief outlook and a discussion on the verifiability of the results that are based on calculations performed by computer programs.

Part I

Background

Chapter 2

Preliminaries

2.1 Basic notation

$\boldsymbol{R}$ is the set of real numbers, $\boldsymbol{Z}$ the set of integers, $\boldsymbol{Z}_p$ the set of integers modulo p, and $\boldsymbol{Z}_p^* = \boldsymbol{Z}_p \setminus \{0\}$. The n-dimensional Euclidean space is denoted $\boldsymbol{R}^n$; the $(n-1)$-dimensional hypersphere in $\boldsymbol{R}^n$ is denoted $\boldsymbol{S}^{n-1}$.

The inner product of two n-dimensional vectors v_1 and v_2 is denoted $\langle v_1, v_2 \rangle$. The norm of a vector v is denoted $\|v\|$; it satisfies $\|v\| = \langle v, v \rangle^{1/2}$.

A regular k-simplex is a set of k unit-length vectors in $\boldsymbol{R}^{k-1}$ such that all pairwise inner products are equal to $-1/(k-1)$.

Boolean OR (disjunction) of two logical variables x and y is denoted $x \vee y$, boolean AND (conjunction) is $x \wedge y$, boolean XOR (exclusive OR) is $x \oplus y$. The negation of the boolean variable x is $\neg x$. A literal is a boolean variable or a negated boolean variable. A boolean formula Φ is in conjunctive normal form (CNF) if it is the conjunction of a set of clauses where each clause is either a literal or the disjunction of two or more literals.

The sign function $\operatorname{sgn} x$ is defined as follows:

$$\operatorname{sgn} x = \begin{cases} +1 & \text{if } x > 0, \\ 0 & \text{if } x = 0, \\ -1 & \text{if } x < 0. \end{cases}$$

2.2 Basic probability theory

For an event A, the complement is denoted $\overline{A}$ while the probability of the event is denoted $\Pr[A]$. For a random variable X, the expected value (or mean) and variance are denoted $\mathrm{E}[X]$ and $\mathrm{Var}[X]$ respectively. The standard deviation of X is denoted σ_X.

The following two theorems connect these entities.

Theorem 2.1 (Markov's inequality). *Let X be a random variable only assuming non-negative values. For all $t > 0$,*

$$\Pr\bigl[X \geq t\bigr] \leq \frac{\mathrm{E}[X]}{t}.$$

Theorem 2.2 (Chebyshev's inequality). *Let X be a random variable with expected value μ and standard deviation σ. For all $t > 0$,*

$$\Pr[|X - \mu| \geq t\sigma] \leq \frac{1}{t^2}.$$

Let $\{A_i\}$ be a set of events and $\{X_i\}$ be a set of random variables. Then

$$\Pr\bigl[\cup A_i\bigr] \leq \sum \Pr[A_i],$$
$$\mathrm{E}\bigl[\sum X_i\bigr] = \sum \mathrm{E}[X_i].$$

2.3 Running time of algorithms

In all analyses of running times of algorithms considered in this thesis, the standard unit-cost model is assumed. The running time is measured as a function of the size of the input to the algorithm in some reasonable encoding. For graph algorithms, it is typically given as a function of the number of vertices in the graph.

When expressing asymptotic relations, e.g. for time complexity, the following standard notation will be used:

- $f(n) \in O(g(n))$ if there exist $c, N > 0$ such that $f(n) < cg(n)$ for all $n > N$.
- $f(n) \in \Omega(g(n))$ if there exist $c, N > 0$ such that $f(n) > cg(n)$ for all $n > N$.
- $f(n) \in \Theta(g(n))$ if there exist $c_1, c_2, N > 0$ such that $c_1 g(n) < f(n) < c_2 g(n)$ for all $n > N$.

2.4 Approximability

Many **NP**-complete decision problems have natural optimization versions; some as maximization problems, and some as minimization problems. These two classes of problems behave in different ways when it comes to the existence of approximation algorithms. All new results in this thesis are for maximization problems.

Definition 2.3. An ***NP** optimization problem* P is characterized by the quadruple $(I_P, SOL_P, m_P, \mathrm{goal}_P)$ where

1. I_P is the set of instances of P. It must be recognizable in polynomial time.

2. SOL_P is a function that maps an input instance x onto the set of feasible solutions for x. There must exist a polynomial $q(x)$ such that for any $y \in SOL_P(x)$, $|y| \leq q(|x|)$. Furthermore, for any y such that $|y| \leq q(x)$, it must be decidable in polynomial time whether $y \in SOL_P(x)$.

3. m_P is a polynomially computable measure function which maps a pair (x, y), where $x \in I_P$ and $y \in SOL_P(x)$, onto the value of y.

4. $\text{goal}_P \in \{\text{MIN}, \text{MAX}\}$ specifies whether P is a minimization or maximization problem.

Solving an **NP** maximization problem P, given an instance x, means finding a feasible solution y such that the objective function value $m_P(x, y)$ is maximized. We denote the optimal value $\max_y m_P(x, y)$ by $\text{opt}_P(x)$. When it is clear from the context what problem is being considered, the subscript will often be omitted.

Definition 2.4. Let x be an instance of an **NP** maximization problem P and let $\text{opt}_P(x)$ be its optimum value. For any solution y to x, the *performance guarantee* is defined as $g_P(x, y) = m_P(x, y)/\operatorname{opt}_P(x)$.

Definition 2.5. An approximation algorithm A for an **NP** maximization problem P has *performance guarantee* $g < 1$ and *performance ratio* $1/g$ if, for all input instances x, $g_P(x, A(x)) \geq g$.

The definition of performance ratio is extended to minimization problems by instead considering the ratio $\text{opt}_P(x)/m_P(x, y)$; this way it is at least 1 for all problems. An approximation algorithm for P with performance guarantee g is often referred to as a g-approximation algorithm, or an algorithm approximating P within g.

Definitions 2.4 and 2.5 can be extended to define expected performance guarantee for randomized approximation algorithms: $m_P(x, y)$ and $m_P(x, A(x))$ are replaced by their expected values $\mathrm{E}[m_P(x, y)]$ and $\mathrm{E}[m_P(x, A(x))]$ respectively.

We are only interested in approximation algorithms that run in polynomial time; even when the running time is not mentioned explicitly, this will be assumed throughout the thesis.

Definition 2.6. The class **Apx** is the class of **NP** optimization problems for which there exist approximation algorithms with constant performance ratios.

Some problems in **Apx** are approximable within *any* constant:

Definition 2.7. A *polynomial time approximation scheme* for a maximization problem P with objective function $m_P(\cdot)$ is a family $\{A_\varepsilon\}$, $\varepsilon > 0$, of algorithms with polynomial running time (for fixed ε) such that $m_P(A_\varepsilon(I)) \geq (1 - \varepsilon)\operatorname{opt}_P(I)$ for all instances I of P.

One can also consider *randomized* polynomial time approximation schemes, for which $m(A_{\varepsilon,\delta}(I)) \geq (1-\varepsilon)\operatorname{opt}(I)$ holds with probability at least $1-\delta$ for all instances I. Note that the running time, although polynomial in the size of the instance, can depend on ε and δ.

Definition 2.8. **PTAS** is the class of **NP** optimization problems for which there exist polynomial-time approximation schemes.

Clearly $\mathbf{PTAS} \subseteq \mathbf{Apx}$, and it was shown by Arora et al. [14] that the inclusion is strict unless $\mathbf{P} = \mathbf{NP}$.

Definition 2.9. **Max-SNP** is the class of **NP** optimization problems that can be written in the form

$$\max_S \bigl|\{x : \Phi(I, S, x)\}\bigr|,$$

where Φ is a quantifier-free formula, I an instance and S a solution. (This class is called $\mathbf{Max\text{-}SNP_0}$ in [72].)

This definition was inspired by Fagin's characterization of **NP** [30]. It is general enough for **Max-SNP** to contain many natural problems. For instance, the Max Cut problem is in **Max-SNP** since it can be expressed as

$$\max_{S \subseteq V} |\{(x,y) : E(x,y) \wedge S(x) \wedge \neg S(y)\}|,$$

where $E(x,y)$ is true if there is an edge (x,y) in the graph and thus plays the role of I in the definition above.

Theorem 2.10. *[74] For every problem $P \in$ **Max-SNP** there exists a constant $c_p > 0$ such that there is a c_p-approximation algorithm for P.*

Theorem 2.11. *[14] If the problem P is **Max-SNP**-complete, then there exists a constant $\varepsilon > 0$ such that there does not exist a $(1-\varepsilon)$-approximation algorithm for P unless $\boldsymbol{P} = \boldsymbol{NP}$.*

A problem P is **Max-SNP**-complete if it is in **Max-SNP** and there exist L-reductions (defined in [74]) from all **Max-SNP** problems to P.

Thus, all **Max-SNP**-complete problems have the property that they can be approximated within some constant factor, but there does not exist a polynomial time approximation scheme unless $\mathbf{P} = \mathbf{NP}$. An example of a **Max-SNP**-complete problem is Max Cut.

2.5 Problems

2.5.1 Constraint satisfaction problems

Most problems considered in this thesis belong to the family of constraint satisfaction problems. We will use the notation of Trevisan et al. [81].

Definition 2.12. A *constraint function of arity* k over $\boldsymbol{Z}_p$ is a function $f : \boldsymbol{Z}_p^k \to \{0, 1\}$.

Definition 2.13. A *constraint family* over $\boldsymbol{Z}_p$ is a finite collection of constraint functions over $\boldsymbol{Z}_p$.

Definition 2.14. A *constraint* C on the variables $x_1, \ldots, x_n$ over a constraint family $\mathcal{F}$ is a $(k+1)$-tuple $(f, i_1, \ldots, i_k)$ where $f \in \mathcal{F}$ has arity k and $i_1, \ldots, i_k$ are distinct integers in $\{1, \ldots, n\}$. The value of the constraint for an assignment $\overline{a}$ to x is $C(a_{i_1}, \ldots, a_{i_k}) = f(x_{i_1}, \ldots, x_{i_k})$ where a value of 1 corresponds to the constraint being satisfied.

Definition 2.15. For a constraint family $\mathcal{F}$ over $\boldsymbol{Z}_p$, the *constraint satisfaction problem MAX($\mathcal{F}$)* is defined as follows: An instance consists of m constraints $C_1, \ldots, C_m \in \mathcal{F}$ on n variables $x_1, \ldots, x_n \in \boldsymbol{Z}_p$. Each constraint C_i has a non-negative weight w_i. The objective is to find an assignment $a_1, \ldots, a_n$ to the variables such that $\sum_{i=1}^m w_i C_i(\overline{a})$ is maximized.

2.5.2 Specific problems

Definition 2.16. The *Max Cut* problem is that of finding a partition of the vertices of a graph $G = (V, E)$ with weights w_{ij} associated with the edges into two subsets so as to maximize the total weight of the edges cut by the partition.

Max Cut can be cast as a binary CSP as follows: There are $n = |V|$ variables $x_1, \ldots, x_n$ corresponding to the vertices of the graph. Each edge (v_i, v_j) corresponds to the constraint $x_i \oplus x_j$ with weight w_{ij}.

Max Cut is a special case of Max p-Cut:

Definition 2.17. The *Max p-Cut* problem is that of finding a partition of the vertices of a graph $G = (V, E)$ with weights w_{ij} associated with the edges into p subsets so as to maximize the total weight of the edges cut by the partition.

In Max p-Cut, the subsets formed need not be balanced, i.e., some subsets may contain more vertices than others. In the problem Max p-Section, this is not allowed:

Definition 2.18. The *Max p-Section* problem is that of finding a partition of the vertices of a graph $G = (V, E)$ with weights w_{ij} associated with the edges into p subsets of equal size so as to maximize the total weight of the edges cut by the partition. For a solution to exist, p must divide $|V|$.

The special case when $p = 2$ is called Max Bisection.

Definition 2.19. The *Max Ek-Sat* problem takes as input a set of boolean variables $x_1, \ldots, x_n$ and a collection of clauses $C_1, \ldots, C_m$ with non-negative weights $w_1, \ldots, w_m$. Each clause is a disjunction of exactly k variables. The objective is to find a truth assignment that maximizes the total weight of satisfied clauses.

We will also encounter the variation Max k-Sat where we also allow clauses that contain less than k variables.

A Horn clause contains at most one negated variable. The Max k-Horn Sat problem is the same as the Max k-Sat problem restricted to instances where all clauses are Horn clauses.

Definition 2.20. Let S be any finite set, $\{S_j\}_{j=1}^n$ be a collection of subsets of S, and $\{w_j\}_{j=1}^n$ be a collection of positive weights corresponding to each subset respectively. The *Max Set Splitting* problem is that of finding a partition of S into two sets that maximizes the total weight of the subsets S_j that are split by the partition.

The Max Ek-Set Splitting problem is the restriction of the Max Set Splitting problem to instances where each subset contains exactly k elements.

Definition 2.21. Let $\{x_i\}_{i=1}^m$ be a collection of boolean variables, $\{C_j\}_{j=1}^n$ be a collection of CNF clauses over those variables, and $\{w_j\}_{j=1}^n$ be a collection of positive weights corresponding to each clause respectively. The *Max NAE Sat* problem is that of finding a truth assignment to the boolean variables that maximizes the total weight of the clauses that contain both true and false literals.

Note that Max Set Splitting is the special case of Max NAE Sat where no negations are allowed.

A large part of this thesis deals with linear equations over $\boldsymbol{Z}_p$. This non-binary CSP is defined as follows:

Definition 2.22. We denote by *Max Ek-Lin mod p* the problem in which the input consists of a system of linear equations mod p in n variables. Each equation contains exactly k variables. The objective is to find the assignment maximizing the number of satisfied equations.

In the variation Max k-Lin mod p we allow equations with less than k variables in each equation.

The linear functions are the most basic functions over $\boldsymbol{Z}_p$. By allowing any function instead of just linear functions, Max Ek-Lin mod p can be generalized to the following problem:

Definition 2.23. We denote by *Max Ek-Function Sat mod p* the problem in which the input consists of a number of functions $\boldsymbol{Z}_p^k \mapsto \boldsymbol{Z}_p$ in n variables. A function is satisfied if it evaluates to zero. The objective is to find the assignment maximizing the number of satisfied functions.

Chapter 3

Randomized approximation algorithms and relaxations

In this chapter we will give a brief introduction to the field of randomized approximation algorithms, focusing on the techniques used in the rest of the thesis.

3.1 Trivial approximation algorithms

For some problems, a very unsophisticated approach can result in an approximation algorithm with a provable performance guarantee. As an example, we will consider the Max E3-Sat problem. An instance of this problem is a set of clauses where each clause is a disjunction of exactly three literals, e.g. $x \vee y \vee \neg z$. A clause must contain three distinct variables. The objective is to find a truth assignment that maximizes the number of satisfied clauses. Let m be the number of clauses.

Consider the following approach: For each logical variable, let it be true with probability $1/2$ and false with probability $1/2$. All random choices are made independently. What is the performance of this extremely simple scheme, which does not use the structure of the clauses? Consider a clause C. There are $2^3 = 8$ different truth assignments to the literals in the clause, each with probability $1/8$ of being chosen by the algorithm. Of these 8 assignments, only 1 will result in the clause not being satisfied. We conclude that the probability that the clause is satisfied is $7/8$. This holds for all the m clauses, and because of the linearity of expectation, the expected number of clauses satisfied by a random assignment is $7m/8$. Obviously the maximum number of clauses that can be satisfied by any assignment is at most m, the total number of clauses. Therefore the expected performance guarantee is at least $7/8$.

The above approximation algorithm uses randomization, so for a particular random choice the actual number of satisfied clauses can be much less than $7m/8$. Running the algorithm several times and taking the best assignment found reduces the probability of this happening. Another approach is to convert the above algorithm into a deterministic approximation algorithm. This can be done using the method of conditional expectation (see e.g. [3]).

As a general technique, the naive approximation algorithm "pick a solution at random from the solution space" is applicable for many problems, and for many maximization problems it leads to good approximation algorithms. In fact, the approximation algorithm for Max E3-Sat described above is the best possible; no approximation algorithm can approximate the optimum within $7/8+\varepsilon$ for any $\varepsilon > 0$ unless $\mathbf{P} = \mathbf{NP}$ [52]. For several other problems, such as Max E2-Sat and Max Cut, it was only recently, with the advent of algorithms based on semidefinite programming, that the naive randomized algorithm was surpassed. A central theme in this thesis is to investigate for what problems there exist approximation algorithms better than picking a solution at random.

3.2 Linear programming relaxations

Consider the Min Vertex Cover problem. An instance of this problem is an undirected graph $G = (V, E)$ where each vertex v_i has a non-negative weight w_i. The objective is to select a subset U of the vertices such that every edge contains at least one vertex from U and the sum of the weights of vertices in U is minimal. Let us now introduce a 0/1-variable x_i corresponding to each vertex v_i, with the meaning that $x_i = 1$ corresponds to $v_i \in U$ and $x_i = 0$ corresponds to $v_i \notin U$. We can now formulate Min Vertex Cover as the following integer program:

$$\begin{array}{lll} \text{minimize} & \displaystyle\sum_i w_i x_i & \\ \text{subject to} & x_i + x_j \geq 1 & \text{for all edges } (v_i, v_j) \in E, \\ & x_i \in \{0,1\} & \text{for all } i. \end{array} \tag{3.1}$$

No efficient algorithm is known for solving integer programs, and there cannot exist any unless $\mathbf{P} = \mathbf{NP}$, so the above formulation does not help us solve the Min Vertex Cover problem. It does however guide us to a natural relaxation which helps us find a good approximation algorithm. Let us relax the last constraint:

$$\begin{array}{lll} \text{minimize} & \displaystyle\sum_i w_i y_i & \\ \text{subject to} & y_i + y_j \geq 1 & \text{for all edges } (v_i, v_j) \in E, \\ & 0 \leq y_i \leq 1 & \text{for all } y_i. \end{array} \tag{3.2}$$

This linear programming (LP) relaxation can be solved in polynomial time, either using Khaciyan's ellipsoid method [64] or some interior point method based on the ideas of Karmarkar [60]. In practice, the algorithms most commonly used are variations of Dantzig's simplex algorithm [27], see any textbook on optimization, even though it requires exponential time in the worst case. Denote the optimum values for (3.1) and (3.2) by opt_{IP} and opt_{LP} respectively. As (3.2) is a relaxation of (3.1), the relation $\text{opt}_{LP} \leq \text{opt}_{IP}$ holds.

Having solved the LP (3.2), we are now faced with the problem of converting the optimal solution into a good solution to the Min Vertex Cover problem. If the optimal LP solution only contains integral values, which means that it is feasible for the IP (3.1), then we have a feasible (and even optimal) solution to the Min Vertex Cover problem. In general, this is not the case, and we have to devise a *rounding scheme* which converts the fractional LP solution to an integral solution, which hopefully has an objective function value, i.e., weight of the vertex cover, close to the LP solution.

Let y^* denote the optimal solution to the LP (3.2). Consider the following rounding scheme:

$$x_i = \begin{cases} 1 & \text{if } y_i^* \geq 0.5, \\ 0 & \text{if } y_i^* < 0.5. \end{cases}$$

That this gives a feasible solution to (3.1) can be seen as follows: Consider an inequality $x_i + x_j \geq 1$ of (3.1). The optimal solution y^* satisfies the inequality $y_i^* + y_j^* \geq 1$. This implies that $\max\{y_i^*, y_j^*\} \geq 0.5$, and the rounding scheme will therefore set at least one of x_i and x_j to 1; hence $x_i + x_j \geq 1$.

It remains to analyze the objective function value. A consequence of the rounding scheme is that $x_i \leq 2y_i^*$ for all i. The weight of the rounded solution therefore satisfies

$$\sum_i w_i x_i \leq \sum_i w_i 2y_i^* = 2\text{opt}_{LP} \leq 2\text{opt}_{IP}.$$

This means that the solution we obtain by solving the LP relaxation (3.2) and applying the above rounding scheme gives a solution to the Min Vertex Cover problem with weight at most twice that of the optimal solution. In the terminology of Chapter 2 we have an approximation algorithm with performance ratio 2.

This approximation algorithm for Min Vertex Cover is due to Hochbaum [46]. A faster approximation algorithm, in which no linear program has to be solved, was constructed by Bar-Yehuda and Even [20]. Both these approximation algorithms have performance ratio 2, and this is in fact the best known for the Min Vertex Cover problem in spite of large efforts. The best non-approximability result, by Håstad [52], is that it is **NP**-hard to achieve performance ratio $7/6 - \varepsilon$ for any $\varepsilon > 0$.

3.3 Semidefinite programming relaxations

Relaxations of integer programs to linear programs have been used for constructions of good approximation algorithms for many problems, but there are problems for which they are not useful. A natural approach is therefore to look for stronger relaxations, hopefully still solvable in polynomial time. However, it was only recently that other relaxations than linear programs have started to become common in the construction of approximation algorithms. In a breakthrough result by Goemans and Williamson [40], it was demonstrated how relaxations to semidefinite programs could be used in order to find better approximation algorithms for a number of problems. Below, their approximation algorithm for the Max Cut problem is described.

3.3.1 Goemans-Williamson's Max Cut relaxation

Given an undirected graph $G = (V, E)$ with n vertices, which we for simplicity will refer to as the integers $1, \dots, n$, where each edge (i, j) has a non-negative weight w_{ij}. The maximum cut can be found by solving the following quadratic integer program:

$$\begin{array}{ll} \text{maximize} & \displaystyle\sum_{i<j} w_{ij} \frac{1 - x_i x_j}{2} \\ \text{subject to} & x_i \in \{-1, 1\} \qquad \forall i. \end{array} \tag{3.3}$$

The two sets in the cut $(S, \overline{S})$ are formed by letting $S = \{i \in V \mid x_i = 1\}$.

While (3.3) is a very neat way to state the Max Cut problem, it is unfortunately hard to solve; all variables are integers, and the objective function is non-linear. With the objective function being non-linear, the hope of finding a useful LP relaxation appears to be small. The key insight of Goemans and Williamson was to replace each x_i with a vector $v_i \in \boldsymbol{R}^n$ of unit length. The product $x_i x_j$ in the objective function is then replaced by the inner product $\langle v_i, v_j \rangle$, leading to the following relaxation:

$$\begin{array}{ll} \text{maximize} & \displaystyle\sum_{i<j} w_{ij} \frac{1 - \langle v_i, v_j \rangle}{2} \\ \text{subject to} & v_i \in \boldsymbol{S}^{n-1} \qquad \forall i. \end{array} \tag{3.4}$$

This is a semidefinite program; we defer the discussion of how to solve it in polynomial time to Section 3.3.4. We denote the optimum to the quadratic integer program (3.3) by opt_{QIP} and the optimum to the semidefinite program (3.4) by opt_{SDP}. As the latter program is a relaxation of the former, we know that $\text{opt}_{SDP} \geq \text{opt}_{QIP}$ holds.

3.3.2 Randomized rounding

Having solved the semidefinite program (3.4), we are now faced with the problem of converting the optimal solution to a good integer solution. In Section 3.2 above, we saw how a fractional solution to the LP relaxation of Min Vertex Cover could be converted into an integer solution using a deterministic rounding scheme. For the Max Cut problem it turns out to be easier to construct a randomized rounding scheme. Let v^* be the optimal solution to (3.4). Choose a vector r randomly with the uniform distribution from the unit sphere $\boldsymbol{S}^{n-1}$. We can now construct a feasible solution to (3.3) through

$$x_i = \begin{cases} +1 & \text{if } \langle v_i^*, r\rangle \geq 0, \\ -1 & \text{if } \langle v_i^*, r\rangle < 0. \end{cases} \tag{3.5}$$

One can also define the cut $(S, \overline{S})$ directly through $S = \{i \in V \mid \langle v_i^*, r\rangle \geq 0\}$. (What happens when $\langle v_i^*, r\rangle = 0$ really does not matter as the probability of this event is 0; we arbitrarily chose to set $x_i = 1$ if that were to happen.)

We will next turn to analyzing the performance of the rounding scheme when applied to the optimal solution to the SDP (3.4). Let W denote the weight of the cut so obtained. As the rounding scheme is randomized, we will analyze the expected value $\mathrm{E}[W]$. To characterize this expected value, we look at each term in the objective function separately. Consider the term $\frac{1-x_i x_j}{2}$, omitting its weight w_{ij}. The corresponding term in the SDP relaxation is $\frac{1-\langle v_i, v_j\rangle}{2}$. What is the probability that the vertices i and j are placed on different sides of the cut? This happens when the vectors v_i and v_j are separated by the hyperplane with normal vector r, an event which occurs with probability θ/π where $\theta = \arccos\langle v_i, v_j\rangle$. Applying this relation to each term in the objective function, we get

$$\mathrm{E}[W] = \frac{1}{\pi}\sum_{i<j} w_{ij} \arccos\langle v_i, v_j\rangle.$$

If we can find a bound of the form

$$\frac{\mathrm{E}[W]}{\mathrm{opt}_{SDP}} = \frac{\frac{1}{\pi}\sum_{i<j} w_{ij} \arccos\langle v_i, v_j\rangle}{\sum_{i<j} w_{ij}\frac{1-v_i^* v_j^*}{2}} \geq \alpha, \tag{3.6}$$

we have shown that the rounding scheme achieves the (expected) performance guarantee α. This follows from

$$\mathrm{E}[W] \geq \alpha \mathrm{opt}_{SDP} \geq \alpha \mathrm{opt}_{QIP}$$

as (3.4) is a relaxation of (3.3). The terms of the sums in the numerators and denominators of (3.6) are in obvious correspondence with each other, and as they are non-negative, it suffices to find a term-wise bound. This means finding α such that

$$\alpha = \min_{v_i, v_j \in \boldsymbol{S}^{n-1}} \frac{2\arccos\langle v_i, v_j\rangle}{\pi(1 - \langle v_i, v_j\rangle)}.$$

This ratio only depends on the inner product $\langle v_i, v_j \rangle$, so we make the substitution $\theta = \arccos\langle v_i, v_j \rangle$ and arrive at the expression

$$\alpha = \min_{\theta \in [0,\pi]} \frac{2\theta}{\pi(1-\cos\theta)}.$$

Using calculus, it is easy to show that the minimum is $\alpha \approx 0.87856$ and that it is attained when $\theta \approx 2.33112$.

The conclusion of this derivation is that the rounding scheme (3.5) when applied to the optimal solution to the SDP relaxation (3.4) gives an approximation algorithm with expected performance guarantee 0.87856 for the Max Cut problem. That the above analysis is tight was shown by Karloff [58]. The above algorithm was first presented in [39]; before that the best known approximation algorithm, due to Sahni and Gonzalez [76], had performance guarantee 0.5.

3.3.3 Max 2-Sat and more sophisticated rounding schemes

The ideas behind the Max Cut relaxation are fairly general, and Goemans and Williamson applied them to a number of other problems. One of those was the Max 2-Sat problem. The input to this problem is a set of disjunctive clauses containing at most two variables, but we will in what follows only describe what happens for clauses containing exactly two variables — it is easy to generalize the algorithm to also handle clauses containing a single variable.

Let $x_1, \dots, x_n$ be integer variables in $\{-1, +1\}$ that correspond to the boolean variables. We introduce new variables $x_{n+1}, \dots, x_{2n}$ such that $x_{n+i} = -x_i$. This is a straightforward way of handling negated variables. We also add a new variable x_0. Variables x_i for which $x_i = x_0$ are interpreted as being false, the others as being true.

$$\begin{array}{lll}
\text{maximize} & \sum_{i,j} w_{ij} z_{ij} & \\
\text{subject to} & z_{ij} \le \frac{3 - x_0 x_i - x_0 x_j - x_i x_j}{4} & \forall i, j, \\
& z_{ij} \le 1 & \forall i, j, \\
& x_{n+i} = -x_i & \forall i, \\
& x_i \in \{-1, 1\} & \forall i, \\
& z_{ij} \in \{0, 1\} & \forall i, j.
\end{array} \tag{3.7}$$

The variable z_{ij} is 1 if the clause containing the variables x_i and x_j is satisfied and 0 otherwise. Note that i and/or j can be larger than n; this is how clauses containing negations are handled. As an example, the variable $z_{n+i,j}$ corresponds to the clause where x_i is negated and x_j is not, i.e., $\neg x_i \vee x_j$.

The above formulation of Max 2-Sat as an integer quadratic program follows the notation of Karloff and Zwick [59] and Zwick [86]; these papers describe a canonical way of generating good relaxations of binary constraint satisfaction problems. It is however easily seen to be equivalent to Goemans-Williamson's original Max 2-Sat formulation.

Just like for Max Cut, we can create a relaxation by expanding the domain of the x_i variables from $\{-1, 1\}$ to $\boldsymbol{S}^n$. This gives the following semidefinite program:

$$\begin{array}{lll} \text{maximize} & \displaystyle\sum_{i,j} w_{ij} z_{ij} & \\ \text{subject to} & z_{ij} \leq \frac{3-\langle v_0,v_i\rangle-\langle v_0,v_j\rangle-\langle v_i,v_j\rangle}{4} & \forall i,j, \\ & z_{ij} \leq 1 & \forall i,j, \\ & v_{n+i} = -v_i & \forall i, \\ & v_i \in \boldsymbol{S}^n & \forall i, \\ & 0 \leq z_{ij} \leq 1 & \forall i,j. \end{array} \tag{3.8}$$

Note that the z_{ij} are relaxed as well, and their range is now $[0, 1]$. This corresponds to clauses being fractionally satisfied.

A slight variation of the hyperplane rounding for Max Cut can be used to convert the optimal solution to the semidefinite program to a good solution to the Max 2-Sat instance. We begin by choosing a vector r randomly with the uniform distribution from the unit sphere $\boldsymbol{S}^n$. An assignment to the logical variables can be constructed using the following rule:

$$x_i = \begin{cases} \text{true} & \text{if } \operatorname{sgn}\langle v_0, r\rangle \neq \operatorname{sgn}\langle v_i, r\rangle, \\ \text{false} & \text{if } \operatorname{sgn}\langle v_0, r\rangle = \operatorname{sgn}\langle v_i, r\rangle. \end{cases} \tag{3.9}$$

This simply means that the variables for which the corresponding vectors are on the same side as v_0 of the hyperplane with normal r through the origin are assigned the logical value "false", while those on the other side are assigned the logical value "true". It can be shown that this rounding scheme gives an approximation algorithm with the same performance guarantee, 0.87856, as the Max Cut algorithm described in Sections 3.3.1 and 3.3.2.

Feige and Goemans [31] came up with a variation of the above rounding scheme. Their key observation was that it might be better to use other vectors than the v_i in (3.9). Each v_i is therefore transformed into a vector v_i' that is coplanar with v_i and v_0 and only depends on these two vectors. The rounding scheme then proceeds to operate on the v_i' instead of the v_i. Specifically, if the angle between v_i and v_0 is θ_i, then the angle between v_i' and v_0 is $f(\theta_i)$ for some function $f(\cdot)$, the *pre-rounding rotation function*. The restriction $f(\pi - \theta) = \pi - f(\theta)$ is added. It guarantees that negated and unnegated variables are treated in the same way. The question is now what function f should be chosen. There is no known analytical expression for the performance guarantee of the rounding scheme as a function of f, so one has to resort to numerical methods for evaluating a particular function f. This can be done by discretizing the set of possible angles between the vectors v_0, v_i and v_j; there are three degrees of freedom for a configuration as all that matters is the angles between pairs of vectors.

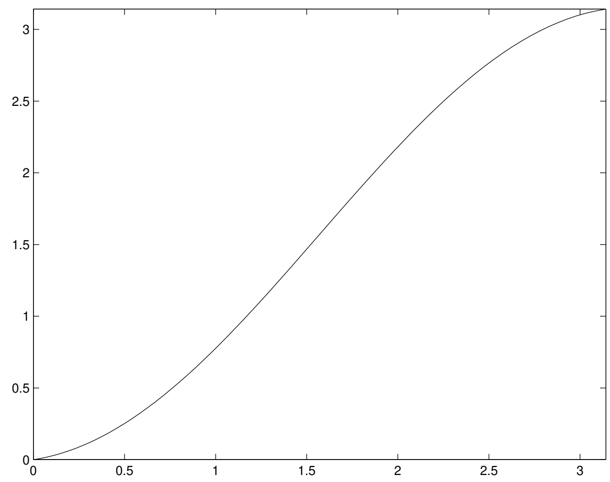

Figure 3.1. Feige-Goemans rotation function.

A good function can be found by trying different functions, either by hand or automatically, and picking the best. Feige and Goemans settled on the function $f(\theta) = \theta + 0.806765(\frac{\pi}{2}(1-\cos\theta)-\theta)$, depicted in Figure 3.1. It gives the performance guarantee 0.93109 for the Max 2-Sat problem, an improvement over the 0.87856 of Goemans and Williamson.

This shows how relatively small modifications of the solution to the semidefinite program can result in a considerable improvement of the performance guarantee. The difficulty lies in the choice of the rotation function f. When only three vectors are involved, it is an easy matter to evaluate the performance numerically, but for some problems one has to deal with four or more vectors. Finding a good rotation function then becomes time-consuming; the infinite space of functions on $[0, \pi]$ can be discretized by only considering piecewise linear functions, but the resulting optimization problem is still very difficult.

Halperin and Zwick [45] describe generalizations of the idea behind pre-rounding rotations and also look into the problem of finding the best rotation functions. They study the Max 4-Sat problem and construct a 0.8721-approximation algorithm using a very general model for how pre-rounding rotations can be used; they select a good rounding scheme from this family using numerical methods. Because of Håstad's lower bound for Max 3-Sat [52], there cannot exist an approximation algorithm for Max 4-Sat with performance guarantee greater than $7/8 = 0.875$. Halperin and Zwick come very close to reaching this goal, but they also prove that their techniques cannot give a 7/8-approximation algorithm without major changes. The lower bound 7/8 also applies to the Max Sat problem, and it has been conjectured that a 7/8-approximation algorithm exists for this problem; the best algorithm for this problem has performance guarantee 0.7846 [17].

3.3.4 Solving the relaxation

Consider the relaxation (3.4). Why is this a semidefinite program? An $n \times n$-matrix A is positive semidefinite if $x^T A x \geq 0$ for all $x \in \boldsymbol{R}^n$. It can be shown (see e.g. [43]) that for a symmetric matrix, this holds if and only if there exists another matrix B such that $A = B^T B$, where B is an $m \times n$-matrix for some $m \leq n$. Given a symmetric $n \times n$-matrix A, such a B can be found in $O(n^3)$ time using incomplete Cholesky decomposition. For a symmetric matrix A such that all diagonal elements are one, the relation $A = B^T B$ can be interpreted as follows: A corresponds to n vectors $v_1, \ldots, v_n \in \boldsymbol{S}^{m-1}$. This follows from letting the vector v_i correspond to the ith column of B, and letting the elements of A satisfy $a_{ij} = \langle v_i, v_j \rangle$. We can now reformulate (3.4) as the semidefinite program

$$\begin{array}{ll} \text{maximize} & \displaystyle\sum_{i<j} w_{ij} \frac{1 - a_{ij}}{2} \\ \text{subject to} & a_{ii} = 1 \quad \forall i \\ & A \text{ symmetric positive semidefinite.} \end{array} \tag{3.10}$$

The optimum solution to a semidefinite program might be irrational, so there is no hope of always finding the solution in polynomial time. It is however possible to find a solution within ε of the optimum value in time polynomial in n and $1/\varepsilon$ for any $\varepsilon > 0$, and this suffices for most applications of semidefinite programming in approximation algorithms. The best methods for solving semidefinite programs, see e.g. [2, 82], are interior-point methods working in the same vein as Karmarkar's polynomial-time algorithm for linear programs [60].

3.3.5 Derandomization

The rounding scheme for Max Cut uses randomization. Is this an intrinsic property of the algorithm, or is it possible to find a deterministic approximation algorithm with the same performance? Goemans and Williamson [40] proposed a simple derandomization scheme, but it was soon discovered to be faulty. Mahajan and Ramesh [70] later presented a derandomization scheme for approximation algorithms based on semidefinite programming, thus providing derandomization of Goemans and Williamson's Max Cut algorithm as well as other algorithms. This result may be interesting from a purely theoretical point of view, but in practice one would instead reduce the probability of finding a bad solution by running the rounding scheme a number of times and picking the best cut so obtained. Even for a large number of repetitions, this is far more efficient than actually running the derandomized version of the algorithm. The reason for this is that Mahajan-Ramesh's derandomization scheme is very complicated and has a huge running time.

3.4 Connection to the rest of the thesis

Semidefinite programming with randomized rounding schemes is used extensively throughout the thesis; Chapters 5–8 all contain approximation algorithms based on this paradigm. Pre-rounding rotations like those described in Section 3.3.3 are used in Chapter 6.

Chapter 4

Non-approximability results

As a complement to the search for better approximation algorithms for various problems, proofs of non-approximability have also been constructed. The first such proofs were typically based on direct combinatorial properties of the problem at hand. Finding such properties was hard, and for most problems no inapproximability results were known. In the last ten years, an advanced mathematical framework has been developed which has produced many new lower bounds. For some problems tight lower bounds have been proved; i.e., lower bounds that match the performances of the best known approximation algorithms.

A lower bound for a particular problem P is typically stated as "if P could be approximated within $f(n)$, then $\mathbf{P} = \mathbf{NP}$" where $f(n)$ is some function, in this thesis always a constant, of the size of the instance of P. For some lower bounds, assumptions stronger than $\mathbf{P} \neq \mathbf{NP}$ are made.

4.1 Probabilistic proof systems

In the early 90s, advances in the theory of probabilistic proof systems produced lower bounds using techniques from coding theory. The first results in this field that had an impact on the field of approximation algorithms are due to Arora et al. [14] building on the results of Arora and Safra [16].

Definition 4.1. A language L is in $\mathbf{PCP}(f(n), g(n))$ if there is a polynomial-time randomized algorithm $V^{\Pi}(r, x)$ that works as follows:

1. It takes as input a string x of length n and a random string of length $O(f(n))$.
2. It generates a query set $Q(r, x) = \{q_1, \dots, q_m\}$ of size $m = O(g(n))$.
3. It reads the bits $\Pi_{q_1}, \dots, \Pi_{q_m}$.
4. It makes a polynomial-time computation using the data r, x and the proof bits $\Pi_{q_1}, \dots, \Pi_{q_m}$ and outputs $V^{\Pi}(r, x) \in \{0, 1\}$.

Moreover, the following acceptance conditions hold for some $\delta > 0$ and for all x:

1. If $x \in L$ then there exists a Π such that for every r we have $M^{\Pi}(r, x) = 1$.
2. If $x \notin L$ then for every Π we have $\Pr_r\big[M^{\Pi}(r, x) = 0\big] \geq \delta$.

The intuition is that there exists a certain proof format that makes it possible to check the proof in just a few places and still be able to discard strings not contained in the language with probability δ.

Theorem 4.2. *[14]* $\boldsymbol{NP} = \boldsymbol{PCP}(\log n, 1)$.

This is a remarkable result; in order to verify (probabilistically) that $x \in L$ where $L \in \mathbf{NP}$, it suffices to read only a constant number of bits from the proof! This highly unnatural property can be achieved by coding the proof Π using a special error correcting code; each bit of a correct proof depends on a large fraction of the bits in a certificate of the fact that $x \in L$.

Corollary 4.3. *[14] There exists a constant $c < 1$ such that it for all $\varepsilon > 0$ is **NP**-hard to approximate Max E3-Sat within $c + \varepsilon$.*

The constant c in the proof in [14] is very close to 1, but in a sequence of improvements of the PCP constructions it has since been reduced. For the purposes of inapproximability results, the best PCP is that of Håstad [52]. He showed the following central result:

Theorem 4.4. *[52] For all $\varepsilon > 0$ it is **NP**-hard to approximate Max E3-Lin mod 2 within a factor $1/2 + \varepsilon$.*

This result is tight as a simple greedy algorithm satisfies at least half the total weight of all equations and therefore has performance guarantee at least $1/2$.

Corollary 4.5. *[52] For all $\varepsilon > 0$ it is **NP**-hard to approximate Max E3-Sat within a factor $7/8 + \varepsilon$.*

This result is also tight as the derandomized version of the naive randomized algorithm (see Section 3.1) has performance guarantee $7/8$.

A note on notation: When referring to inapproximability results such as Corollary 4.5, the qualifier "For all $\varepsilon > 0$" will often be omitted in running text. Furthermore, all inapproximability results from now on are based on the assumption $\mathbf{P} \neq \mathbf{NP}$. This will also be implicitly assumed in a number of places. In particular, Theorem 4.5 would be referred to as stating that Max E3-Sat cannot be approximated within $7/8 + \varepsilon$.

The following theorem, implicit in [21], has been a source of a large number of inapproximability results. It provides a connection between the tight inapproximability result of Theorem 4.4 and the gadget construction techniques of Trevisan et al. [81] (described in the next section).

Definition 4.6. The constraint families PC_0 and PC_1 (PC for Parity Check) are defined through the relations

$$\mathrm{PC}_i(x,y,z) = \begin{cases} 1 & \text{if } x+y+z=i, \\ 0 & \text{otherwise.} \end{cases}$$

Theorem 4.7. *[21] If there exists an α_0-gadget reducing PC_0 to $\mathcal{F}$ and an α_1-gadget reducing PC_1 to $\mathcal{F}$, then for all $\varepsilon > 0$, $MAX(\mathcal{F})$ is hard to approximate within $1 - \frac{1}{\alpha_0+\alpha_1} + \varepsilon$.*

For many problems this theorem gives the best lower bounds currently known, often close to the performances of the best approximation algorithms.

4.2 Finding optimal gadget reductions

In some cases it is possible to transfer a lower bound for one problem P_1 to a lower bound for another problem P_2. This is done using combinatorial reductions between the two problems, much in the same spirit as the reductions used to prove **NP**-completeness. What lower bound one gets for P_2 depends on how the reduction is designed. Many, if not most, reductions are based on local replacement rules: A substructure of an instance of P_1 is transformed into another substructure, a gadget, of an instance of P_2. These local rules are easier to construct and analyze than more global constructions. Until recently, there was no good measure of quality for gadget reductions. Bellare, Goldreich and Sudan [21] proposed the following definition which remedied this deficiency:

Definition 4.8. For $\alpha \in \boldsymbol{R}$, a binary constraint function f of arity k and a binary constraint family $\mathcal{F}$, an *α-gadget* reducing f to $\mathcal{F}$ is a set of variables $y_1, \ldots, y_n$, a finite collection of real weights $w_j \geq 0$, and associated constraints C_j from $\mathcal{F}$ over primary variables $x_1, \ldots, x_k$ and auxiliary variables $y_1, \ldots, y_n$, with the property that, for boolean assignments $\overline{a}$ to $x_1, \ldots, x_k$ and $\overline{b}$ to $y_1, \ldots, y_n$, the following are satisfied:

$$(\forall \overline{a} : f(\overline{a}) = 1)(\forall \overline{b}) : \sum_j w_j C_j(\overline{a}, \overline{b}) \leq \alpha,$$
$$(\forall \overline{a} : f(\overline{a}) = 1)(\exists \overline{b}) : \sum_j w_j C_j(\overline{a}, \overline{b}) = \alpha,$$
$$(\forall \overline{a} : f(\overline{a}) = 0)(\forall \overline{b}) : \sum_j w_j C_j(\overline{a}, \overline{b}) \leq \alpha - 1.$$

The gadget is strict if, in addition,

$$(\forall \overline{a} : f(\overline{a}) = 0)(\exists \overline{b}) : \sum_j w_j C_j(\overline{a}, \overline{b}) = \alpha - 1.$$

α is the cost of the gadget; a smaller α gives a better lower bound for $\mathrm{MAX}(\mathcal{F})$.

This definition formalized arguments about gadgets, but the problem of finding good gadgets remained an ad hoc task. The ultimate goal would be to find an optimal gadget and prove its optimality, but this was very hard to do by hand. Trevisan et al. [81] showed that for a large class of problems, constraint satisfaction problems of the form $\mathrm{MAX}(\mathcal{F})$ where $\mathcal{F}$ is a hereditary constraint family, an optimal gadget can be found by solving a linear program. Many optimal gadgets have since been found using this technique. The following exposition describes the key definitions and theorems from that paper. For the sake of simplicity, all results are stated for binary constraint functions. They can easily be generalized to hold for constraint functions over $\boldsymbol{Z}_p$.

Definition 4.9. A constraint family $\mathcal{F}$ is *hereditary* if for any $f \in \mathcal{F}$ of arity k and any two indices $i, j \in [k]$, the function f when restricted to assignments such that $x_i = x_j$ and considered as a function of $k-1$ variables is identical to some other function $f' \in \mathcal{F} \cup \{0, 1\}$.

$\{0\}$ and $\{1\}$ are the all-zero and all-one functions, respectively.

Definition 4.10. For a constraint f, two variables a_i and a_j are *distinct* if there exists an assignment $\overline{a}$, satisfying f, for which $a_i \neq a_j$.

Theorem 4.11. *[81] Let f be a constraint function of arity k with s satisfying assignments. Let $\mathcal{F}$ be a constraint family and $\alpha \geq 1$ be such that there exists an α-gadget reducing f to $\mathcal{F}$. If $\mathcal{F}$ is hereditary then there exists an α'-gadget with at most $2^s - k'$ auxiliary variables reducing f to $\mathcal{F}$, where $\alpha' \leq \alpha$, and k' is the number of distinct variables among the satisfying assignments of f.*

Theorem 4.11 restricts the search for optimal gadgets to gadgets with a bounded number of auxiliary variables. This is an interesting result in itself, but Trevisan et al. went on to prove an even stronger result: When Theorem 4.11 applies, an optimal gadget can be found by solving a linear program.

Definition 4.12. Consider a constraint function f of arity k with s satisfying assignments and a hereditary constraint family $\mathcal{F}$. The $(f, \mathcal{F})$*-canonical witness matrix* is an $s \times (2^s + k - k')$-matrix where all elements are 0 or 1. The first k columns contain all the satisfying assignments of f (one per row) and the remaining columns contain all possible columns that are distinct from each other and from the columns corresponding to the primary variables.

A witness matrix M induces a natural mapping $b_M : \{a \in \{0,1\}^k \mid f(a) = 1\} \to \{0,1\}^{2^s - k'}$ by looking at each row separately. The intuition here is that this mapping specifies the values of the auxiliary variables for each satisfying assignment.

Definition 4.13. Consider a constraint function f of arity k and a constraint family $\mathcal{F}$. Let M be the $(f, \mathcal{F})$-canonical witness matrix and b_M be the induced mapping. Let $C_1, \ldots, C_m$ be all the possible constraints that can be created by applying a constraint from $\mathcal{F}$ to a set of $k+n$ boolean variables. Thus for every j, $C_j : \{0,1\}^{k+n} \to \{0,1\}$. Let $w_1, \dots, w_m$ be the weights corresponding to the constraints and α be the cost of the gadget. The linear program LP$(f, \mathcal{F}, M, n)$ is the following:

$$\begin{array}{lll} \text{minimize} & \alpha & \\ \text{subject to} & (\forall \overline{a} : f(\overline{a}) = 1) : & \sum_{i=1}^{m} w_i C_i(\overline{a}, b_M(\overline{a})) = \alpha, \\ & (\forall \overline{a} : f(\overline{a}) = 1)(\forall \overline{b}) : & \sum_{i=1}^{m} w_i C_i(\overline{a}, \overline{b}) \leq \alpha, \\ & (\forall \overline{a} : f(\overline{a}) = 0)(\forall \overline{b}) : & \sum_{i=1}^{m} w_i C_i(\overline{a}, \overline{b}) \leq \alpha - 1, \\ & w_i \geq 0 \quad \forall i. & \end{array} \tag{4.1}$$

Theorem 4.14. *[81] Let f be a constraint function of arity k with s satisfying assignments and $\mathcal{F}$ be a hereditary constraint family. Then an optimal gadget reducing f to MAX($\mathcal{F}$) can be found by solving LP$(f, \mathcal{F}, M, n)$ for $n = 2^s - k'$ where k' is the number of distinct variables among the satisfying assignments of f.*

Note that the bound $2^s - k'$ on the number of auxiliary variables in Theorems 4.11 and 4.14 is an upper bound; for constraint families $\mathcal{F}$ exhibiting symmetries, the bound can be lowered. In particular, for constraint families that are symmetric with respect to negation, at most $2^{s-1} - k'$ auxiliary variables are needed.

The gadget construction method outlined above only works for weighted problems — unweighted problems would result in integer programs. For most problems, this does not matter as Crescenzi, Silvestri and Trevisan [25] have shown that the weighted and unweighted versions of a large class of problems are equally hard to approximate. This holds for all problems considered in this thesis.

The feasibility of this approach for finding optimal gadgets depends heavily on the number of satisfying assignments for the source function f. The LP (4.1) has $m+1$ variables, where m depends on $\mathcal{F}$, and $2^{2^s+k-k'} + s$ inequalities. This doubly exponential dependency on s makes this approach infeasible unless s is small. In view of the inapproximability results on Håstad, and especially Theorem 4.7, the most natural functions to reduce from are PC_0 and PC_1, both having $s = 4$. The resulting LPs are large but not intractably so for most target families $\mathcal{F}$ of interest.

4.3 Connection to the rest of the thesis

In Chapter 6, a variation of the gadget construction techniques described in Section 4.2 is used to find provably optimal gadgets as well as strong lower bounds. In Chapter 7, a hand-crafted gadget is used to derive an inapproximability result.

Part II

New results

Chapter 5

An approximation algorithm for Max Set Splitting

5.1 Introduction

In this chapter, we introduce new approximation algorithms for Max Set Splitting and Max Not-All-Equal Sat. In the Max Set Splitting problem, a problem instance consists of subsets of some finite set. The problem is to partition the elements into two parts, such that as many subsets as possible are split, i.e., contain elements from both parts. A restriction of this problem, Max Ek-Set Splitting, where all subsets have cardinality k, was shown to be **NP**-complete for any fixed k by Lovász [67]. It has also been shown to be **Apx**-complete [75]. Obviously, Max 2-Set Splitting is exactly the Max Cut problem. The best known approximation algorithm for this problem, described in Chapter 3, has performance guarantee 0.87856 [40]. Furthermore, Max 3-Set Splitting has been shown to be approximable with the same performance guarantee [55], and for $k \geq 4$, Max Ek-Set Splitting is approximable within $1-2^{1-k}$ [1, 55]; this can be obtained by derandomizing the simple randomized algorithm. However, the previously best known algorithm for the general Max Set Splitting problem has a performance guarantee of 0.5.

The Max Not-All-Equal Sat problem, from now on abbreviated Max NAE Sat, is a variation of Max Sat, where the goal is to maximize the total weight of the clauses that contain both true and false literals. It has been shown to be **Apx**-complete with performance guarantee 0.5 [74]. We can actually view Max NAE Sat as a generalization of Max Set Splitting, since every instance of the Max Set Splitting problem is also an instance of Max NAE Sat, namely an instance where no clause contains any negated literal.

Our approximation algorithm for Max Set Splitting is built upon the ideas of Crescenzi and Trevisan [26]. We start by solving a semidefinite program obtained from Goemans and Williamson [40], and then add a probabilistic postprocessing step, where the rounded solution to the semidefinite program is perturbed. A small modification of this algorithm also gives an algorithm for Max NAE Sat. By formulating the performance guarantee of this combined algorithm as the objective function of a linear program, we can bound it by simply solving the linear program. This shows that the combined algorithm has a performance guarantee of 0.72405, both for Max Set Splitting and Max NAE Sat.

5.2 The algorithms for Max Set Splitting

5.2.1 The relaxation

Let $u(S_j)$ be a function that is 0 if S_j is not split, and at least 1 if S_j is split. Then we can formulate the Max Set Splitting problem as the integer program

$$\begin{array}{lll} \text{maximize} & \sum_{j=1}^{n} w_j z_j & \\ \text{subject to} & u(S_j) \geq z_j & \forall j, \\ & z_j \in \{0,1\} & \forall j. \end{array}$$

We now aim to find a suitable expression for $u(S_j)$.

Definition 5.1. For each subset S_j, we let $P_j = \big\{\{i_1, i_2\} : i_1 \neq i_2 \wedge i_1, i_2 \in S_j\big\}$.

We observe that none of the pairs $\{i_1, i_2\} \in P_j$ are split if S_j is not split, and that at least $|S_j| - 1$ of the pairs are split if S_j is split. Furthermore, we let $y_i \in \{-1, 1\}$ correspond to the element $x_i \in S$, where $x_{i_1} \in S$ and $x_{i_2} \in S$ belong to the same part in the partition of S if and only if $y_{i_1} = y_{i_2}$. This enables us to define $u(S_j)$ as

$$u(S_j) = \frac{1}{|S_j| - 1} \sum_{\{i_1, i_2\} \in P_j} \frac{1 - y_{i_1} y_{i_2}}{2}.$$

We now introduce $|S|$-dimensional unit vectors v_i instead of the y_i to relax the integer program to a semidefinite one. Each product $y_{i_1} y_{i_2}$ in the definition of $u(S_j)$ is replaced by the inner product $\langle v_{i_1}, v_{i_2} \rangle$. This yields the following semidefinite program:

$$\begin{array}{lll} \text{maximize} & \sum_{j=1}^{n} w_j z_j & \\ \text{subject to} & \frac{1}{|S_j|-1} \sum_{\{i_1, i_2\} \in P_j} \frac{1 - \langle v_{i_1}, v_{i_2} \rangle}{2} \geq z_j & \forall j, \\ & 0 \leq z_j \leq 1 & \forall j, \\ & \langle v_i, v_i \rangle = 1 & \forall j. \end{array} \tag{5.1}$$

Each vector v_i in the solution to the semidefinite program is used to assign a value in $\{-1, 1\}$ to the corresponding y_i by the following randomized rounding scheme:

A random hyperplane through the origin is chosen. Denote its normal by r. Then we set $y_i = \operatorname{sgn}\langle r, v_i\rangle$. This is the randomized rounding scheme introduced by Goemans and Williamson for the Max Cut problem. To analyze the performance of this algorithm we introduce the following indicator random variables:

Definition 5.2. For each pair $\{y_{i_1}, y_{i_2}\}$ of variables, the indicator random variable $X_{i_1 i_2}$ is defined by

$$X_{i_1 i_2} = \begin{cases} 1 & \text{if } y_{i_1} \neq y_{i_2}, \\ 0 & \text{otherwise.} \end{cases}$$

Definition 5.3. For each subset S_j, the indicator random variable Z_j is defined by

$$Z_j = \begin{cases} 1 & \text{if } S_j \text{ is split by the algorithm,} \\ 0 & \text{otherwise.} \end{cases}$$

Our goal now is to show a lower bound on $\mathrm{E}\left[\sum_{j=1}^{n} w_j Z_j\right] = \sum_{j=1}^{n} w_j \mathrm{E}[Z_j]$. To do this, we will use a bound on $\Pr[X_{i_1 i_2} = 1]$ and a relation between Z_j and $\sum_{\{i_1,i_2\}\in P_j} X_{i_1 i_2}$.

We will only have $X_{i_1 i_2} = 1$ if the vectors v_{i_1} and v_{i_2} end up on opposite sides of the random hyperplane. The probability of this event is proportional to the angle between the vectors, and if the vectors are anti-parallel, they will always be on opposite sides of the hyperplane. Thus $\Pr[X_{i_1 i_2} = 1] = \theta_{i_1 i_2}/\pi$, where $\theta_{i_1 i_2}$ is the angle between v_{i_1} and v_{i_2}. Goemans and Williamson showed that this probability can be bounded by

$$\Pr[X_{i_1 i_2} = 1] \geq \alpha \frac{1 - \langle v_{i_1}, v_{i_2}\rangle}{2}, \tag{5.2}$$

where $\alpha > 0.87856$ [40] (see also the discussion in Section 3.3). This result alone takes care of any subset S_j of cardinality 2. For larger subsets we use an idea exploited by Goemans and Williamson [40] for Max Sat:

Lemma 5.4. *Define β_k as*

$$\beta_k = \begin{cases} 4/k^2 & \text{if } k \text{ is even,} \\ 4/(k+1)(k-1) & \text{if } k \text{ is odd.} \end{cases}$$

Then, for any subset S_j, we have that $Z_j \geq \beta_{|S_j|} \sum_{\{i_1,i_2\}\in P_j} X_{i_1 i_2}$.

Proof. For any subset S_j, at most $1/\beta_{|S_j|}$ of the pairs in P_j are split if S_j is split. Also, none of the pairs in P_j are split when S_j is not split. □

Note that if $|S_j| = 3$, exactly $1/\beta_3 = 2$ of the pairs in P_j are split if S_j is split. This fact is the basis for the 0.87856-approximation algorithm for Max 3-Set Splitting due to Kann, Lagergren and Panconesi [55].

Putting the pieces together, we obtain the following bound on $\mathrm{E}[Z_j]$:

Theorem 5.5. *For a subset S_j, we have that* $\mathrm{E}[Z_j] \geq \alpha\gamma_{|S_j|} z_j$, *where*

$$\gamma_{|S_j|} = \begin{cases} 4(|S_j|-1)/|S_j|^2 & \text{if } |S_j| \text{ is even,} \\ 4/(|S_j|+1) & \text{if } |S_j| \text{ is odd.} \end{cases}$$

Proof. By Lemma 5.4 and the linearity of expectation,

$$\mathrm{E}[Z_j] \geq \beta_{|S_j|} \sum_{\{i_1,i_2\}\in P_j} \mathrm{E}[X_{i_1 i_2}].$$

Now, by the bound of Goemans and Williamson from (5.2), $\mathrm{E}[X_{i_1 i_2}] \geq \alpha(1 - \langle v_{i_1}, v_{i_2}\rangle)/2$, since $X_{i_1 i_2}$ is an indicator random variable. Thus,

$$\mathrm{E}[Z_j] \geq \alpha\beta_{|S_j|}\big(|S_j|-1\big)\frac{1}{|S_j|-1} \sum_{\{i_1,i_2\}\in P_j} \frac{1-\langle v_{i_1}, v_{i_2}\rangle}{2}.$$

Applying the first inequality in the semidefinite program (5.1) to the right-hand side, we obtain

$$\mathrm{E}[Z_j] \geq \alpha\beta_{|S_j|}(|S_j|-1)z_j = \alpha\gamma_{|S_j|}z_j$$

where the equality follows from the definitions of β_k and γ_k. □

As Z_j is an indicator variable, $\Pr[Z_j] = \mathrm{E}[Z_j]$. We will use this interpretation of the bound in Section 5.3.

To sum up, we have established that the algorithm produces a solution with an expected weight of at least $\sum_{j=1}^{n} \alpha\gamma_{|S_j|} w_j z_j$.

5.2.2 Random perturbation

We will in the next section analyze the combined algorithm that runs the following algorithms and takes the maximum weight obtained as the result.

Algorithm 1. *For each $x_i \in S$, the part in which x_i is put is chosen randomly; both parts have probability $1/2$ of being chosen. A simple analysis shows that a set S_j is split with probability $1 - 2^{1-|S_j|}$.*

Algorithm 2. *The algorithm from Section 5.2.1.*

Algorithm 2 has severe problems with large subsets. On the other hand, a large set is split with high probability if the partition is chosen at random. We thus aim to combine the benefits from those two approaches without suffering from their drawbacks. Inspired by techniques introduced by Crescenzi and Trevisan [26], we construct the following algorithm:

Algorithm 3. *For a given instance, we start by running the algorithm from Section 5.2.1. Then, we perturb the obtained answer by letting each x_i switch part in the partition with probability p.*

The best value of p will be specified later. Note that if we set $p = 0$ in Algorithm 3, we get Algorithm 2.

5.3 Analyzing the combined algorithm

As we take the maximum result obtained by any of the above algorithms, we want to establish a lower bound on the maximum of the contributing algorithms. Usually, such an analysis has been accomplished by constructing a new combined algorithm, which chooses as its answer the outcome of the ith contributing algorithm with probability q_i. Then the expected value of this new algorithm is calculated. By the linearity of expectation, the calculation can be performed separately for subsets of cardinality k, for each k, thus ensuring that the weakness of one algorithm is compensated by the strength of the other algorithms. The maximum of the solutions from the contributing algorithms can never be smaller than the expected value of the combined algorithm, and we therefore obtain a bound on the performance guarantee. In Section 5.3.2 we will show how to find the best set of q_i by solving a linear program.

5.3.1 Separate analyses of the contributing algorithms

Let A_i denote the weight of the solution obtained by Algorithm i above. We now aim to show lower bounds for A_i.

We first focus on finding a lower bound on $A_3(p)$, Algorithm 3 run with switching probability p. In the analysis we will use the constant N to denote the cut-off point between small and large subsets; these two cases will be handled separately. The value of N will be specified later. The reason why we make this separation is that we want to be able to solve a linear program with one constraint for each possible subset size; to keep the number of constraints finite we have one single constraint for all subset sizes that are at least N. This is reasonable as the probability that a subset S_j is split after the perturbation is high when $|S_j|$ is large and $p > 0$.

Definition 5.6. Let $w_\infty = \sum_{j\,:\,|S_j|\geq N} w_j$ be the total weight of all large subsets.

The bounds obtained will be expressions containing z_j. The z_j are obtained from the solution to the semidefinite program (5.1).

Definition 5.7. Let B_j and F_j denote the events "S_j was split before the perturbation" and "S_j is split after the perturbation" respectively.

Lemma 5.8. $\Pr[F_j \mid B_j] \geq 1 - p(1-p)^{|S_j|-1} - p^{|S_j|-1}(1-p)$.

Proof. Assume that S_j before the perturbation was split into two sets of sizes ℓ and $|S_j| - \ell$ respectively, where $0 < \ell < |S_j|$. Now

$$\Pr[F_j \mid B_j] = 1 - p^{\ell}(1-p)^{|S_j|-\ell} - p^{|S_j|-\ell}(1-p)^{\ell}$$

and using elementary calculus and the fact that $p \leq 1/2$, it can be shown that this expression is minimized when $\ell = 1$ or $\ell = |S_j| - 1$. $\square$

To simplify the notation we make the following definition:

Definition 5.9. For $k < N$, let

$$g_k(p) = \alpha\gamma_k\big(1 - p(1-p)^{k-1} - p^{k-1}(1-p)\big) + (1-\alpha\gamma_k)\big(1 - p^k - (1-p)^k\big).$$

Furthermore, let $g_N(p) = 1 - p^N - (1-p)^N$.

Lemma 5.10. $\Pr[F_j] \geq z_j g_{|S_j|}(p)$ *when* $|S_j| < N$.

Proof. Using Lemma 5.8 and $\Pr[F_j \mid \overline{B}_j] = 1 - p^{|S_j|} - (1-p)^{|S_j|}$, we obtain

$$\begin{aligned}
\Pr[F_j] &= \Pr[F_j \mid B_j]\Pr[B_j] + \Pr[F_j \mid \overline{B}_j]\Pr[\overline{B}_j] \\
&\geq \Pr[B_j]\big(1 - p(1-p)^{|S_j|-1} - p^{|S_j|-1}(1-p)\big) \\
&\qquad + (1 - \Pr[B_j])\big(1 - p^{|S_j|} - (1-p)^{|S_j|}\big) \\
&= \big(1 - p^{|S_j|} - (1-p)^{|S_j|}\big) \\
&\qquad + \Pr[B_j](1-2p)\big((1-p)^{|S_j|-1} - p^{|S_j|-1}\big).
\end{aligned}$$

But $\Pr[B_j] \geq \alpha\gamma_{|S_j|}z_j$ (by Theorem 5.5) and $p \leq 1/2$; hence

$$\begin{aligned}
\Pr[F_j] &\geq \big(1 - p^{|S_j|} - (1-p)^{|S_j|}\big) \\
&\qquad + \alpha\gamma_{|S_j|}z_j(1-2p)\big((1-p)^{|S_j|-1} - p^{|S_j|-1}\big) \\
&= \alpha\gamma_{|S_j|}z_j\big(1 - p(1-p)^{|S_j|-1} - p^{|S_j|-1}(1-p)\big) \\
&\qquad + (1 - \alpha\gamma_{|S_j|}z_j)\big(1 - p^{|S_j|} - (1-p)^{|S_j|}\big) \\
&\geq z_j g_{|S_j|}(p),
\end{aligned}$$

as $z_j \leq 1$ implies $1 - \alpha\gamma_{|S_j|}z_j \geq 1 - \alpha\gamma_{|S_j|} \geq (1 - \alpha\gamma_{|S_j|})z_j$. $\square$

Lemma 5.11. $\Pr[F_j] \geq g_N(p)$ *when* $|S_j| \geq N$.

Proof. We begin by showing that $\Pr[F_j \mid B_j] \geq \Pr[F_j \mid \overline{B_j}]$ for $p \leq 0.5$:

$$\begin{aligned}\Pr[F_j \mid B_j] - \Pr[F_j \mid \overline{B_j}] &\geq p^{|S_j|} - p^{|S_j|-1}(1-p) + (1-p)^{|S_j|} - p(1-p)^{|S_j|-1} \\ &= (1-2p)\big((1-p)^{|S_j|-1} - p^{|S_j|-1}\big) \\ &\geq 0\,,\end{aligned}$$

where the first inequality follows from Lemma 5.8 and the second from the fact that $p \leq 0.5$. The lemma now follows from

$$\Pr[F_j] \geq \Pr[F_j \mid \overline{B}_j] = 1 - p^{|S_j|} - (1-p)^{|S_j|} \geq g_N(p).$$

□

We are now able to formulate a bound on $A_3(p)$:

Theorem 5.12. $A_3(p) \geq \sum_{k=2}^{N-1} \sum_{j:|S_j|=k} w_j z_j g_k(p) + w_\infty g_N(p)$.

Proof. Follows immediately from Lemmas 5.10 and 5.11. □

The bounds for A_1 and A_2 follow from previous work [1, 40, 54, 55]:

$$A_1 \geq \sum_{k=2}^{N-1} \sum_{j:|S_j|=k} w_j(1-2^{1-k}) + w_\infty(1-2^{1-N})$$

and

$$A_2 \geq \sum_{k=2}^{N-1} \sum_{j:|S_j|=k} w_j \alpha \gamma_k z_j\,.$$

5.3.2 The worst case for the best algorithm

To obtain an expression for the expected value of the combination of the above algorithms, we notice that, since $0 \leq z_j \leq 1$, A_1 is at least

$$\sum_{k=2}^{N-1} \sum_{j:|S_j|=k} w_j z_j(1-2^{1-k}) + w_\infty(1-2^{1-N}).$$

When this is combined with Theorem 5.12, it follows that the expected weight of the solution from the combined algorithm described in Section 5.2.2 is at least

$$\sum_{k=2}^{N-1} \sum_{j:|S_j|=k} w_j z_j a_k + w_\infty a_N\,,$$

where

$$a_k = q_{-1}(1 - 2^{1-k}) + \sum_{j=0}^{M} q_j g_k(j/2M). \tag{5.3}$$

Here q_{-1} is the probability that Algorithm 1 is used while q_k, $0 \le k \le M$, is the probability that Algorithm 3 with $p = k/2M$ is used. (Notice that $p = 0$ corresponds to Algorithm 2.) The parameter M, which is used to discretize the probabilities p used in Algorithm 3, will be specified below.

To obtain a bound on the performance guarantee, we solve the following linear program:

$$\begin{array}{lll} \text{maximize}_{q_{-1},q_0,\dots,q_M} & \min_{k\in[2,N]} a_k & \\ \text{subject to} & \sum q_j = 1, & \\ & q_j \ge 0 & \forall j. \end{array} \tag{5.4}$$

The true contribution from the subsets of cardinality at least N will always be larger than what is specified by the constraints. Thus, the minimum obtained from the linear program is a lower bound on the performance guarantee of the algorithm.

To compute the optimum of the linear program, we must select values for M and N. When $M = N = 50$, the optimal value of the linear program is at least 0.72405. This means that the algorithm will always deliver a result which is at least

$$0.72405 \left(\sum_{k=2}^{N-1} \sum_{j:|S_j|=k} w_j z_j + w_\infty \right) \ge 0.72405 \cdot \text{opt},$$

which shows that the algorithm is a 0.72405-approximation algorithm. It turns out that the only non-zero q_k are q_{11} and q_{12}; both these are approximately 0.5. This implies that the best performance is obtained by one single algorithm: Hyperplane rounding is applied to the solution of the relaxation (5.1), and then a perturbation with $p \in [0.11, 0.12]$ is performed.

The term a_N in (5.3) is much greater than 0.9, given the solution to the linear program. This means that the answer obtained from the linear program is unaffected by the fact that we underestimate the contribution from sets of cardinality at least N. If we decrease N below 50 we obtain a slightly lower optimum, while the optimum does not increase if we increase N.

We also varied M, and observed that there was no substantial improvement when we increased M above 50.

To solve the linear programs, we used the publicly available package LPSOLVE by Michel Berkelaar. The running time for $M = N = 50$ on a Sun Ultra 1 workstation was less than one second.

A small modification of the algorithms described above gives a performance guarantee of 0.72405 also for Max NAE Sat: If a variable x_i occurs negated in a clause, the corresponding vector v_i in the corresponding constraint in the semidefinite relaxation is replaced by $-v_i$.

5.4 Lower bounds

The relevant lower bounds for Max Set Splitting and its variations are all derived for versions in which all sets contain the same number of elements. Håstad [52] showed a lower bound of $7/8 + \varepsilon$ for Max E4-Set Splitting. This lower bound is tight, as picking a partition at random gives a performance guarantee of 7/8 for this problem. Håstad in fact showed something stronger: That it is **NP**-hard to distinguish between instances where all sets can be split and instances where at most a fraction $7/8 + \varepsilon$ of the sets can be split, for all $\varepsilon > 0$. In a related result, Guruswami [44] showed a lower bound of $21/22 + \varepsilon$ for Max E3-Set Splitting and a lower bound of $27/28 + \varepsilon$ for satisfiable instances of Max E3-Set Splitting. Håstad's result implies a lower bound of $7/8 + \varepsilon$ for Max Set Splitting, and this is the best lower bound also for Max NAE Sat. These lower bounds can be compared with the performance guarantee of 0.72405 attained by our algorithm.

5.5 Discussion

The ideas of our approximation algorithm are applicable whenever there exists an algorithm that performs well on some class of input instances, and the naive probabilistic algorithm, which simply chooses a feasible solution at random, performs well on some other class of instances. For some problems, it may also be possible to use some information from the solution obtained by the first algorithm to choose the probabilities in the postprocessing step. The drawback is, as is indeed the case for Max Set Splitting and Max NAE Sat, that the probabilistic postprocessing can destroy the good performance of the first algorithm.

The approach of expressing the performance of a combination of approximation algorithms as a linear program is fairly general. Whenever there exist different approximation algorithms, where some algorithms deliver good approximations for one class of instances, and other algorithms deliver good approximations for some other class of instances, our technique may be applicable. A prerequisite of our analysis is that it is possible to somehow express the performance of the algorithms in such a way that it can be related to the optimum. For instance, it seems hard to combine two different algorithms that are based on semidefinite programming. Our framework is, of course, not restricted to maximization problems; it works on minimization problems as well.

5.6 Recent results

The results in this chapter first appeared in [5]. Zwick later constructed a 0.90872-approximation algorithm for Max NAE 3-Sat [88]. This algorithm fits the framework of Section 5.2.2 and the analysis in Section 5.3.2, and a 0.73698-approximation algorithm for Max NAE Sat follows. Zwick goes on to describe an algorithm for Max NAE Sat which he is unable to analyze; if a certain conjecture, supported by some numerical evidence, is true, a 0.7977-approximation algorithm for Max NAE Sat follows. All these results of course carry over to the special case Max Set Splitting.

Chapter 6

On the approximability of Max k-Horn Sat

6.1 Introduction

One of the canonical problems in the theory of **NP**-completeness and approximation algorithms is the Max Sat problem. The best approximation algorithm for this problem is due to Asano and Williamson [17] and has performance guarantee approximately 0.7846. Various restricted versions of the Max Sat problem have also been considered. Several versions in which the number of variables in each clause is fixed have been extensively studied. For the Max 2-Sat problem, Feige and Goemans [31] constructed a 0.93109-approximation algorithm based on a semidefinite programming relaxation and pre-rounding rotations. The best lower bound for this problem is $21/22 + \varepsilon$; this bound follows when the non-approximability results of Håstad [52] are combined with the gadget of Bellare, Goldreich and Sudan [21] (later proven optimal by Trevisan et al. [81]). For the Max E3-Sat problem, the naive randomized algorithm, with performance guarantee $7/8$, was shown to be the best possible by Håstad [52]. Karloff and Zwick used semidefinite programming to construct a $7/8$-approximation algorithm also for the generalization Max 3-Sat [59].

Schaefer [77] categorized the decision versions of boolean satisfiability problems. Somewhat surprisingly, it turns out that all decision problems of the form $\text{SAT}(\mathcal{F})$ are either in **P** or **NP**-complete and, unless $\mathbf{P} = \mathbf{NP}$, the only such problems that can be solved in polynomial time are 2-Sat, Horn Sat and Lin mod 2. Here $\text{SAT}(\mathcal{F})$ is the decision problem whether a collection of boolean constraints from the family $\mathcal{F}$ can all be simultaneously satisfied. In 2-Sat, each constraint is the disjunction of two literals, in Lin mod 2, each constraint is the exclusive OR of one or more literals, and in Horn Sat, each constraint is the disjunction of one or more literals, at most one of which is a negated variable.

The approximability properties of Max 2-Sat have been extensively studied, the best known results are summarized above. The Max Ek-Lin mod 2 problem was recently considered by Håstad [52], who showed that achieving a performance guarantee of $1/2+\varepsilon$ is **NP**-hard. This result is tight as picking a variable assignment at random satisfies half the equations on average.

The maximization version of Horn Sat, the third decision problem classified as "easy" by Schaefer, has been subject to much less study, in spite of it being a relatively natural problem. In this chapter we will investigate the approximability properties of Max Horn Sat when restricted to instances where there is a bound on the number of literals in each clause, considering both lower and upper bounds.

Khanna, Sudan and Williamson [65] performed a categorization of maximization versions of constraint satisfaction problems. An implication of their results is that Max Horn Sat is **Max-SNP**-hard. The only other result on the approximability of the Max Horn Sat problem that we are aware of is due to Zwick [87]. He considered the robustness of approximation algorithms for some problems, and one of his results was that if it is possible to satisfy a fraction $1-\varepsilon$ of the clauses in a Max Horn Sat instance, then there exists a polynomial-time algorithm that finds an assignment which satisfies $1-8\log\log\frac{1}{\varepsilon}/\log\frac{1}{\varepsilon}$ of the clauses.

6.2 An approximation algorithm for Max 3-Horn Sat

6.2.1 Discussion

Max 3-Horn Sat is a special case of Max 3-Sat, so the 7/8-approximation algorithm by Karloff and Zwick [59] has the same performance guarantee also for Max 3-Horn Sat. Lacking dedicated approximation algorithms for Max 3-Horn Sat, this is the best known approximation algorithm also for this problem.

We will base our approximation algorithm for Max 3-Horn Sat on Karloff and Zwick's canonical semidefinite relaxation of the integer program for Max 3-Sat:

$$\begin{array}{lll}
\text{maximize} & \sum_i w_i z_i + \sum_{i,j} w_{ij} z_{ij} + \sum_{i,j,k} w_{ijk} z_{ijk} & \\
\text{subject to} & z_i \le relax(v_0, v_i) & \forall i, \\
& z_{ij} \le relax(v_0, v_i, v_j) & \forall i,j, \\
& z_{ijk} \le relax(v_0, v_i, v_j, v_k) & \forall i,j,k, \\
& v_i \in \boldsymbol{S}^n & \forall i, \\
& v_{n+i} = -v_i & 1 \le i \le n.
\end{array} \tag{6.1}$$

Indices $1,\dots,n$ correspond to the boolean variables present in the Max 3-Sat instance, while indices $n+1,\dots,2n$ correspond to their negations. The variables z_i, z_{ij} and z_{ijk} represent the degrees to which clauses of length one, two and three respectively are satisfied, and the w_i, w_{ij} and w_{ijk} are the corresponding weights.

Weights equal to zero indicate clauses that are not present in the instance. The function *relax* holds the constraints that the solution must satisfy. For clauses containing one or two variables, it essentially reduces to Goemans-Williamson's relaxation:

$$\begin{aligned} relax(v_0, v_i) &= \frac{1 - \langle v_0, v_i \rangle}{2}, \\ relax(v_0, v_i, v_j) &= \min\left\{ \frac{3 - \langle v_0, v_i \rangle - \langle v_0, v_j \rangle - \langle v_i, v_j \rangle}{4}, 1 \right\}. \end{aligned}$$

For clauses of length three, the constraints are more complicated:

$$\begin{aligned} relax(v_0, v_i, v_j, v_k) = \min\Bigg\{ &\frac{4 - \langle v_0 + v_i, v_j + v_k \rangle}{4}, \frac{4 - \langle v_0 + v_j, v_i + v_k \rangle}{4}, \\ &\frac{4 - \langle v_0 + v_k, v_i + v_j \rangle}{4}, 1 \Bigg\}. \end{aligned} \tag{6.2}$$

Karloff and Zwick used the randomized rounding scheme introduced by Goemans and Williamson [40] for the Max 2-Sat problem (see Chapter 3). They achieved the performance guarantee $7/8$, which is the best possible for this problem in the light of the result by Håstad [52]: It is **NP**-hard to approximate Max E3-Sat within $7/8 + \varepsilon$ even when restricted to satisfiable instances. It is easy to analyze the performance of the rounding scheme for clauses of length one and two, but for clauses of length three this becomes considerably more difficult. A clause will be satisfied after the randomized rounding if at least one of the literals is true, which corresponds to at least one of v_i, v_j and v_k not ending up on the same side of the random hyperplane as v_0. We therefore let

$$\begin{aligned} prob(v_0, v_1, \dots, v_k) = \Pr\big[\{v_0, v_1, \dots, v_k\} &\text{ are separated by the} \\ &\text{random hyperplane with normal } r\}\big]. \end{aligned}$$

Analogously to the derivation in Section 3.3, the performance guarantee for clauses of length three is

$$\min_{v_0, v_i, v_j, v_k \in \boldsymbol{S}^n} \frac{prob(v_0, v_i, v_j, v_k)}{relax(v_0, v_i, v_j, v_k)}.$$

We have an explicit formula for the denominator, (6.2), but for the numerator the most explicit expression known is

$$prob(v_0, v_i, v_j, v_k) = 1 - \frac{volume\big(tetra(v_0, v_i, v_j, v_k)\big)}{\pi^2}$$

where $tetra(v_0, v_i, v_j, v_k)$ is the spherical tetrahedron defined as

$$tetra(v_0, v_i, v_j, v_k) = \big\{ r \in \boldsymbol{S}^3 \mid \langle v_0, r \rangle \geq 0, \langle v_i, r \rangle \geq 0, \langle v_j, r \rangle \geq 0, \langle v_k, r \rangle \geq 0 \big\}.$$

There is no closed form for the volume function of spherical tetrahedra, and there is reason to believe that none exists as it is related to the dilogarithm function [63]. One therefore has to resort to numerical integration to evaluate it. Several different integrals representing the volume function have been proposed [23, 24, 49, 78]. The most recent is also the most useful for the purposes of numerical integration. It is due to Hsiang [49] and reduces the problem of evaluating the volume function to calculating a complicated one-dimensional integral.

6.2.2 A new rounding scheme

Let us begin by revisiting the approximation algorithm of Feige and Goemans for Max 2-Sat [31] (described in Chapter 3). When restricted to clauses of length at most two, the relaxation (6.1) above is equivalent to Feige-Goemans' relaxation (3.8). This means that the only difference in how such clauses are handled is the rounding scheme. Feige-Goemans' pre-rounding rotations achieve a performance guarantee of 0.93109. This is considerably more than the 0.87856 that Karloff-Zwick's algorithm achieves for clauses of length at most two, but on the other hand there is no point in applying pre-rounding rotations to the optimal solution of (6.1) as an optimal approximation result is achieved for Max 3-Sat anyway.

When we turn to Max 3-Horn Sat, the bound in Theorem 4.5 does not apply. A pre-rounding rotation can improve the performance for clauses of length two; if we can find one that also provides improvement for the two possible types of 3-Horn clauses, $x \vee y \vee z$ and $x \vee y \vee \neg z$, then we can approximate the Max 3-Horn Sat problem better than the 7/8 achieved by Karloff-Zwick's general Max 3-Sat algorithm.

The pre-rounding rotations considered so far, see e.g. [45, 59, 86], all have the symmetry property $f(\pi - \theta) = \pi - f(\theta)$ to ensure that negated and unnegated variables are treated in the same way. When we consider the Max 3-Horn Sat problem, it is no longer obvious that such a symmetry is beneficial. This is because of the lack of symmetry of Max 3-Horn Sat instances — a clause of length three contains at most one negated variable. The worst case for Karloff-Zwick's Max 3-Sat algorithm is when the four vectors v_0, v_i, v_j, v_k are pairwise orthogonal. For such a configuration it is easy to see that $relax(v_0, v_i, v_j, v_k) = 1$ and, looking at Hsiang's formulas [49], $prob(v_0, v_i, v_j, v_k) = 7/8$. It is clear that the pre-rounding functions used by Feige and Goemans can never improve on this configuration as the symmetry property implies that $f(\pi/2) = \pi/2$. This means that the vectors v_i, v_j and v_k are not affected by the rotation. Let us now ignore the symmetry property and suppose that $f(\pi/2) = \pi/2 + \varepsilon$ for some small $\varepsilon > 0$. Those among v_i, v_j and v_k that correspond to unnegated variables will now be rounded away from v_0, and the probability that such literals are satisfied after the randomized rounding is greater than 1/2. The drawback is of course that those corresponding to negated variables are rounded towards v_0, making the probability of the literal being satisfied less than 1/2. For a Max 3-Horn Sat clause containing three variables, the probability of it being satisfied should nevertheless increase as there are at least two literals for which the probability of being true increases, and at most one for which the probability decreases. This indicates that it might prove fruitful to apply pre-rounding rotations without the symmetry property in order to construct a better approximation algorithm for Max 3-Horn Sat.

Finding a good rotation function is a non-trivial optimization problem. Like Halperin and Zwick [45], we restricted the attention to continuous piecewise linear functions $f : [0, \pi] \to [0, \pi]$ such that $f(0) = 0$ and $f(\pi) = \pi$. A relatively small number of bends, typically between 5 and 15, was used when searching for good rotation functions. For a particular function f, the performance was determined by discretizing the six angles that describe a configuration of v_0, v_i, v_j and v_k and numerically calculating $prob(v_0, v_i, v_j, v_k)$ using Hsiang's formula. (The performance for clauses containing one or two variables was also determined, but this was considerably easier as no numerical integration was necessary in these cases.) Each angle was divided into up to 60 steps.

To prune the angle space, only configurations satisfying the Feige-Goemans inequalities

$$\begin{aligned}
-\langle v_0, v_i\rangle + \langle v_0, v_j\rangle + \langle v_i, v_j\rangle &\leq 1\\
+\langle v_0, v_i\rangle - \langle v_0, v_j\rangle + \langle v_i, v_j\rangle &\leq 1\\
+\langle v_0, v_i\rangle + \langle v_0, v_j\rangle - \langle v_i, v_j\rangle &\leq 1\\
-\langle v_0, v_i\rangle - \langle v_0, v_j\rangle - \langle v_i, v_j\rangle &\leq 1\\
-\langle v_0, v_i\rangle + \langle v_0, v_k\rangle + \langle v_i, v_k\rangle &\leq 1\\
+\langle v_0, v_i\rangle - \langle v_0, v_k\rangle + \langle v_i, v_k\rangle &\leq 1\\
+\langle v_0, v_i\rangle + \langle v_0, v_k\rangle - \langle v_i, v_k\rangle &\leq 1\\
-\langle v_0, v_i\rangle - \langle v_0, v_k\rangle - \langle v_i, v_k\rangle &\leq 1\\
-\langle v_0, v_j\rangle + \langle v_0, v_k\rangle + \langle v_j, v_k\rangle &\leq 1\\
+\langle v_0, v_j\rangle - \langle v_0, v_k\rangle + \langle v_j, v_k\rangle &\leq 1\\
+\langle v_0, v_j\rangle + \langle v_0, v_k\rangle - \langle v_j, v_k\rangle &\leq 1\\
-\langle v_0, v_j\rangle - \langle v_0, v_k\rangle - \langle v_j, v_k\rangle &\leq 1
\end{aligned}$$

were considered. These inequalities are part of the polytope that defines Max 3-Sat (see [59, 86] for a discussion on canonical relaxations) and can therefore be included in the relaxation. Another way to see that they can be safely included is that they are valid in an integral solution, i.e., when all v_i are ± 1.

The best function discovered in the search for good rotation functions was a rather gentle rotation function; $|f(\theta) - \theta|$ is at most 0.12. This is attained for $\theta = \pi/2$, and just as expected it pays off to rotate all such vectors away from v_0. The function is depicted in Figure 6.1 and tabulated in Table 6.1. One can also note that it shares some properties with Feige-Goemans rotation function (see Figure 3.1): For angles in the neighborhood of 0.81 and 2.33, the worst cases for Goemans-Williamson's Max 2-Sat algorithm, it rotates the angles towards 0 and π respectively. Although this is not the worst-case configuration for Karloff-Zwick's

Max 3-Sat algorithm, such rotations are still necessary as it otherwise would be impossible to achieve a performance guarantee larger than that of Goemans and Williamson's Max 2-Sat algorithm, 0.87856.

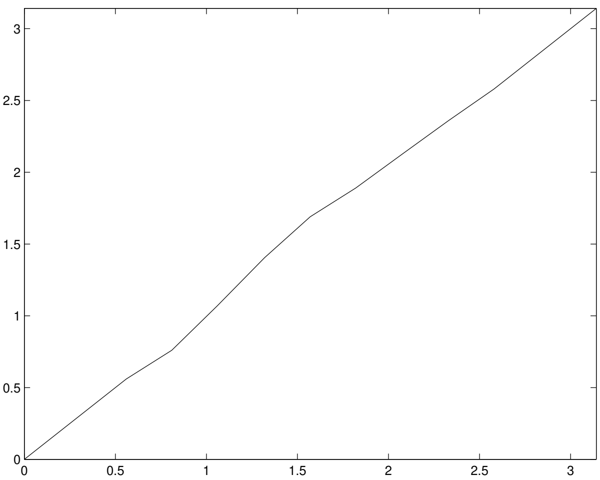

Figure 6.1. Rotation function for Max 3-Horn Sat.

Θ	$f(\Theta)$	Θ	$f(\Theta)$
0	0	1.82	1.89
0.56	0.56	2.08	2.13
0.81	0.76	2.33	2.36
1.06	1.07	2.58	2.58
1.32	1.405	π	π
$\pi/2$	1.69		

Table 6.1. Rotation function for Max 3-Horn Sat.

Conjecture 6.1. *The performance guarantee of the Max 3-Horn Sat algorithm that solves the relaxation (6.1), rotates the vectors according to the function tabulated in Table 6.1, and then performs hyperplane rounding, is 0.882.*

This result rests upon numerical evidence rather than an analytical proof. We verified that the result holds for all discretizations of the angle space into $20^6, 21^6, \ldots, 59^6, 60^6$ points. Finding an analytical proof by hand seems out of the question; it might be possible to generate a proof with the help of a computer algebra system. Even if this could be done, it is doubtful how much insight such a proof would give — it would probably be very long, tedious and hard to understand.

Karloff and Zwick [59] were initially not able to prove that their algorithm gives a performance guarantee of 7/8 for Max 3-Sat, but were later able to construct a (complicated) proof. The situation looks even worse for Max 3-Horn Sat as a rotation function is involved. The recent results by Halperin and Zwick for Max 4-Sat [45] are also based on numerical evidence.

The above Max 3-Horn Sat algorithm gives a rather small improvement in the performance guarantee over the Max 3-Sat algorithm by Karloff and Zwick, 0.882 vs 0.875. We therefore searched for more complicated rounding schemes, but without success Halperin and Zwick report similar experiences in the full version of [45]; they studied some variations of Max Sat, and in instances not containing any clauses of length more than three, the best rounding schemes were plain pre-rounding rotations combined with hyperplane rounding. Picking a random truth assignment gives a performance guarantee of 0.875 for clauses containing exactly three variables, and this indicates that rounding schemes that use more randomness can hurt the performance. The worst-case configurations for Goemans-Williamson's Max 2-Sat algorithm also come into play as the performance guarantee of that algorithm is only 0.87856. For the current 0.882-approximation algorithm there are several different configurations for which the worst case is attained. Some of these are low-dimensional configurations, close to the worst-case configurations for the Max 2-Sat algorithms, and some are configurations for which v_0, v_i, v_j and v_k are almost orthogonal.

For Max 2-Horn Sat, using asymmetric rotation functions did not improve on the 0.93109-approximation algorithm of Feige and Goemans for Max 2-Sat. For Max 4-Horn Sat, the rotation function in Table 6.1 gives a performance guarantee of about 0.86, significantly worse than the 0.8721-approximation algorithm for general Max 4-Sat by Halperin and Zwick. They used a more complicated rounding scheme, and this might be necessary also for Max 4-Horn Sat.

6.3 Lower bounds

For some problems in the Max k-Horn Sat family, we prove lower bounds using the methodology of Trevisan et al. [81] (see Section 4.2). The constraint family Max k-Horn Sat is hereditary for any k, and we have therefore been able to find optimal gadgets reducing PC_0 and PC_1 to Max 2-Horn Sat, Max 3-Horn Sat and Max 4-Horn Sat.

The performances of the new gadgets and the lower bounds they give are listed in Section 6.3.1. A discussion on the adaptations of the methodology of Trevisan et al. [81] that were necessary for the LPs to become tractable follows in Section 6.3.2. The gadgets obtained are listed in Section 6.3.3.

6.3.1 New inapproximability results

Lemma 6.2. *There exists a 13-gadget reducing PC_0 to 2-Horn and it is optimal.*

Lemma 6.3. *There exists a 12-gadget reducing PC_1 to 2-Horn and it is optimal.*

Both these gadgets contain four auxiliary variables, which is the minimal number of auxiliary variables for any optimal such gadget.

Theorem 6.4. *For all $\varepsilon > 0$, it is **NP**-hard to approximate Max 2-Horn Sat within $24/25 + \varepsilon$.*

Proof. Follows from Lemmas 6.2 and 6.3 and Theorem 4.7. ☐

This can be compared with the lower bound of $21/22 + \varepsilon$ for the Max 2-Sat problem.

Lemma 6.5. *There exists a 7-gadget reducing PC_0 to 3-Horn and it is optimal and strict.*

Lemma 6.6. *There exists a 7-gadget reducing PC_1 to 3-Horn and it is optimal.*

The PC_0-gadget does not use any auxiliary variable while the PC_1-gadget uses three auxiliary variables.

Theorem 6.7. *For all $\varepsilon > 0$, it is **NP**-hard to approximate Max 3-Horn Sat within $13/14 + \varepsilon$.*

Proof. Follows from Lemmas 6.5 and 6.6 and Theorem 4.7. ☐

The corresponding lower bound for Max 3-Sat is $7/8 + \varepsilon$.

Lemma 6.8. *There exists a 7-gadget reducing PC_0 to 4-Horn and it is optimal and strict.*

Lemma 6.9. *There exists a 7-gadget reducing PC_1 to 4-Horn and it is optimal.*

This implies that we cannot get a stronger lower bound for Max 4-Horn Sat than Theorem 6.7 by considering gadget reductions from the PC family.

6.3.2 Methodology

We are interested in gadgets from PC_0 and PC_1, and both these constraint functions have 4 satisfying assignments. By Theorem 4.11, we need only consider gadgets containing at most $2^4 - 3 = 13$ auxiliary variables. This can be reduced to 11 by observing that setting a variable identically false or identically true in a Horn clause results in another Horn clause. On the other hand, Max k-Horn Sat is not symmetric with respect to negation as a Horn clause cannot contain more than one negated variable. With this symmetry, $2^{4-1} - 3 = 5$ auxiliary variables would suffice, but it is unfortunately not present in the target functions we consider. This leaves us with 3 primary and 11 auxiliary variables for all the gadgets we consider. The LP (4.1) will therefore contain $2^{14} + 4 = 16388$ constraints, while the number of variables depends on what member of the Max k-Horn Sat family we choose as target for the reduction. By counting the number of Horn clauses that can be formed using 14 different variables, we can calculate the sizes of the LPs we have to solve in order to find optimal gadgets from PC_i to Max 2-Horn Sat, Max 3-Horn Sat and Max 4-Horn Sat. They are listed in Table 6.2. The number

Problem	Variables	Inequalities
Max 2-Horn Sat	302	16388
Max 3-Horn Sat	1758	16388
Max 4-Horn Sat	6763	16388

Table 6.2. Dimensions of the LPs for finding optimal gadgets.

of inequalities is much larger than the number of variables for all the LPs, so it turned out to be easier to solve the dual LPs rather than the original LPs. We tried to use the software package LPSOLVE to solve the LPs, but only succeeded in solving the LPs corresponding to the two Max 2-Horn Sat gadgets — the others were too large for our hardware/software configuration. What makes these LPs cumbersome is that almost all elements in the constraint matrices are non-zero. In most applications where large LPs are solved, the constraint matrices are extremely sparse, thus greatly simplifying the task for the LP solver. Furthermore, many LP packages, including LPSOLVE, are optimized for sparse LPs, resulting in bad memory management when applied to dense LPs.

To overcome this difficulty, we considered relaxations of (4.1). Most optimal gadget reductions from PC_0 and PC_1 to target families of interest contain a small number of auxiliary variables, usually much less than the upper bound provided by Theorem 4.11. We can exploit this fact by only considering a subset B of the possible assignments to the auxiliary variables $\overline{b}$ in the LP (4.1). Consider the relaxation

$$\begin{array}{lll} \text{minimize} & \alpha & \\ \text{subject to} & (\forall \overline{a} : f(\overline{a}) = 1) : & \sum_{j=1}^{m} w_j C_j(\overline{a}, b_M(\overline{a})) = \alpha, \\ & (\forall \overline{a} : f(\overline{a}) = 1)(\forall \overline{b} \in B) : & \sum_{j=1}^{m} w_j C_j(\overline{a}, \overline{b}) \leq \alpha, \\ & (\forall \overline{a} : f(\overline{a}) = 0)(\forall \overline{b} \in B) : & \sum_{j=1}^{m} w_j C_j(\overline{a}, \overline{b}) \leq \alpha - 1, \\ & w_i \geq 0 \quad \forall i. & \end{array} \tag{6.3}$$

Let v be the number of auxiliary variables that we guess will be present in the optimal gadget. Then we choose B such that for each of the $\binom{11}{v}$ v-subsets of the auxiliary variables, all 2^v assignments occur at least once in B. We also want B to be small for the relaxation to be significantly faster to solve than the original LP. To accomplish this, we formulated the problem of finding a minimal B as an instance of the Min Set Cover problem and applied the well-known greedy approximation algorithm (see e.g. [47]). This approach was good enough for finding small sets B, and we did not look into more sophisticated techniques.

By introducing the parameter v in the construction of B, we get a nice parameterization of the search for gadgets, as the size of the relaxed LP (6.3) is an increasing function of v. We therefore generated and solved the LPs for different values of v. For each value of v, the optimal solution to (6.3) was inserted into the original LP (4.1) and checked for feasibility — when the solution to the relaxed LP (6.3) is feasible for the original LP (4.1), it must, being a relaxation, also be optimal. Using this technique, we could find provably optimal gadget reductions by solving LPs much smaller than the bounds from Table 6.2. The actual sizes of the LPs that resulted in optimal gadgets are listed in Table 6.3.

Gadget	v	Variables	Inequalities
$\mathrm{PC}_0 \to$ Max 2-Horn Sat	7	302	2812
$\mathrm{PC}_1 \to$ Max 2-Horn Sat	7	302	2812
$\mathrm{PC}_0 \to$ Max 3-Horn Sat	6	1758	1508
$\mathrm{PC}_1 \to$ Max 3-Horn Sat	5	1758	708
$\mathrm{PC}_0 \to$ Max 4-Horn Sat	6	6763	1508
$\mathrm{PC}_1 \to$ Max 4-Horn Sat	6	6763	1508

Table 6.3. Dimensions of relaxations solved for finding optimal gadgets.

It turns out that the value of v for which the solution to the relaxation (6.3) becomes feasible for (4.1) is between 5 and 7 for all gadgets whereas the number of auxiliary variables is between 0 and 4. The intuition that v corresponds to the number of auxiliary variables is therefore slightly misleading; a larger value of v is necessary in all cases.

We also verified that the gadgets contain a minimal number of auxiliary variables. This can be achieved by considering restricted witness matrices: Suppose we want to show that there does not exist any gadget with less than 4 auxiliary variables for some reduction. Then it suffices to solve $\binom{11}{3} = 165$ different LPs to verify this; one for each way to choose 3 out of the 11 columns for auxiliary variables in the canonical witness matrix. These LPs are very small and can be solved in less than a second each.

6.3.3 New gadgets

The new gadgets from PC_0 and PC_1 to Max 2-Horn Sat and Max 3-Horn Sat are listed in Tables 6.4–6.7. Notation: The x_i are the variables of the constraint being reduced, while the y_i are auxiliary variables, private for each group of constraints from the target family.

Weight	Constraint	Weight	Constraint
2	$\neg x_3$	1	$x_1 \vee \neg y_4$
1	$\neg y_1$	1	$x_2 \vee \neg y_1$
1	$\neg y_3$	1	$x_2 \vee \neg y_4$
1	$x_2 \vee y_2$	1	$\neg x_1 \vee y_1$
1	$x_2 \vee y_3$	1	$\neg x_1 \vee y_2$
1	$x_3 \vee y_1$	1	$\neg x_3 \vee y_2$
1	$x_3 \vee y_3$	1	$\neg x_3 \vee y_4$
1	$x_1 \vee \neg y_3$		

Table 6.4. The 13-gadget reducing the PC_0 constraint $\neg(x_1 \oplus x_2 \oplus x_3)$ to a set of Max 2-Horn Sat constraints.

Weight	Constraint	Weight	Constraint
1	$\neg x_1$	1	$x_2 \vee \neg y_2$
1	$\neg x_3$	1	$x_3 \vee \neg y_4$
1	$x_1 \vee y_4$	1	$\neg x_1 \vee y_1$
1	$x_2 \vee y_3$	1	$\neg x_1 \vee y_2$
1	$x_3 \vee y_2$	1	$\neg x_2 \vee y_4$
1	$x_1 \vee \neg y_3$	1	$\neg x_3 \vee y_1$
1	$x_2 \vee \neg y_1$	1	$\neg x_3 \vee y_3$

Table 6.5. The 12-gadget reducing the PC_1 constraint $x_1 \oplus x_2 \oplus x_3$ to a set of Max 2-Horn Sat constraints.

Weight	Constraint	Weight	Constraint
1	$\neg x_1$	2	$x_1 \vee x_3$
1	$\neg x_2$	2	$\neg x_1 \vee x_2 \vee x_3$
1	$\neg x_3$	2	$x_1 \vee x_2 \vee \neg x_3$

Table 6.6. The 7-gadget reducing the PC_0 constraint $\neg(x_1 \oplus x_2 \oplus x_3)$ to a set of Max 3-Horn Sat constraints.

Weight	Constraint	Weight	Constraint
1	$x_1 \vee x_2 \vee x_3$	0.5	$x_2 \vee y_2$
0.5	$\neg x_1$	0.5	$x_3 \vee y_1$
0.5	$\neg x_2$	0.5	$x_1 \vee \neg x_2 \vee y_1$
0.5	$\neg x_3$	0.5	$x_1 \vee \neg x_3 \vee y_2$
0.5	$\neg y_1$	0.5	$x_2 \vee \neg x_3 \vee y_3$
0.5	$\neg y_2$	0.5	$x_1 \vee x_2 \vee \neg y_1$
0.5	$\neg y_3$	0.5	$x_1 \vee x_3 \vee \neg y_2$
0.5	$x_1 \vee y_3$	0.5	$x_2 \vee x_3 \vee \neg y_3$

Table 6.7. The 7-gadget reducing the PC_1 constraint $x_1 \oplus x_2 \oplus x_3$ to a set of Max 3-Horn Sat constraints.

Chapter 7

Approximating linear equations mod p

7.1 Introduction

Systems of linear equations mod p is a basic and very general combinatorial problem, which exhibits the following property: The naive randomized algorithm that chooses a solution at random approximates the problem within $1/p$. Recently, Håstad [52] studied systems of linear equations mod p with exactly k unknowns in each equation, and showed that it is **NP**-hard to approximate the problem within $1/p+\varepsilon$ for all $\varepsilon > 0$, all $p \geq 2$, and all $k \geq 3$.

In this chapter we study the only interesting remaining cases, systems of linear equations mod p with at most, or exactly, two unknowns in each equation. When $p = 2$, this problem has been studied previously, but for $p > 2$ not much is known. We use semidefinite programming combined with randomized rounding to show that for both Max 2-Lin mod p and Max E2-Lin mod p it is possible to do better than the naive randomized heuristic. Specifically, we show that there exists, for all p, a randomized polynomial time algorithm that approximates both problems within $1/(1-\kappa(p))p$, where $\kappa(p) > 0$ for all p. On the negative side, we show that it is **NP**-hard to approximate Max E2-Lin mod p within some constant performance guarantee, independent of p.

As described in Chapter 3, the usual way to use semidefinite programming in approximation algorithms is to formulate the problem as an integer program, and then relax this program to a semidefinite one. In order to approximate Max p-Cut, Frieze and Jerrum [36] instead associated a vector with each vertex, and added constraints enforcing the vectors to have certain properties. To refine their technique, we let each variable in the system of linear equations be represented by a constellation of several vectors. By adding suitably chosen constraints to the semidefinite program, we make sure that the solution to the semidefinite program has the same kind of symmetries as the solution to the original problem.

Our approach is in some sense dual to that of Frieze and Jerrum. We use many vectors to represent each variable and one random vector in the rounding; they use one vector for each variable and many random vectors in the rounding. Our algorithm can be used also for Max p-Cut, since Max p-Cut is a special case of Max E2-Lin mod p. It is not clear a priori how our method and the method of Frieze and Jerrum relate to each other. We elucidate on this and show that the performance guarantee of our algorithm obtained by only considering local configurations, the usual method of analysis, is not better than that of Frieze and Jerrum's algorithm.

7.2 Preliminaries

From now on, p always denotes a prime, although most of our results generalize to composite p. Regarding the lower bound, it is easy to see that if p is a prime factor in m we can convert a Max E2-Lin mod p instance to an equivalent Max E2-Lin mod m instance by multiplying each equation with m/p. Since we show a constant lower bound, independent of p, the lower bound generalizes. We will show later how to generalize our upper bounds to composite p.

We now show that a simple randomized heuristic, which can be derandomized by the method of conditional expectation, for Max 2-Lin mod m has performance guarantee $1/m$. Since an equation $ax_i - bx_{i'} = c \bmod m$ can only be satisfied if $\gcd(a, b, m)$ divides c, we can assume that all equations have this property; it suffices to satisfy a fraction $1/m$ of the satisfiable equations.

Algorithm 4. *Takes as input an instance of Max 2-Lin mod m, where $m = p_1^{\alpha_1} \cdots p_k^{\alpha_k}$, with variables $x_1, \ldots, x_n$. Outputs an assignment with expected weight at least a fraction $1/m$ of the weight of the satisfiable equations in the instance. The algorithm guesses, for each j, the values of $x_i \bmod p_j^{\alpha_j}$ uniformly at random.*

By the Chinese remainder theorem, x_i is determined uniquely by the values of $x_i \bmod p_j^{\alpha_j}$ for $j = 1, \ldots, k$.

Lemma 7.1. *If we guess an assignment to the $x_i \bmod p_s^{\alpha_s}$ uniformly at random, a satisfiable equation of the form $ax_i - bx_{i'} = c \bmod p_s^{\alpha_s}$ is satisfied with probability at least $1/p_s^{\alpha_s}$.*

Proof. If either a or b is a unit mod $p_s^{\alpha_s}$, the proof is trivial. Otherwise, $\gcd(a, b) = p^t$ for some $t \geq 1$, and in this case we can divide a, b and c by p^t to produce an equivalent equation

$$\frac{a}{p^t} x_i - \frac{b}{p^t} x_{i'} = \frac{c}{p^t} \bmod p_s^{\alpha_s - t}.$$

(Note that $p^t \mid c$ for the equation to be satisfiable.) This equation will be satisfied with probability greater than $1/p_s^{\alpha_s}$. □

Corollary 7.2. *There exists, for all $m \geq 2$, a deterministic algorithm for Max 2-Lin mod m with performance guarantee $1/m$.*

Proof. By Lemma 7.1, Algorithm 4 satisfies any satisfiable equation with probability at least $1/m$. With this in mind, the corollary follows from the facts that the optimum of an instance is at most the weight of the satisfiable equations, and that the algorithm can be derandomized by the standard technique of conditional expectation. In this method one determines the values of the variables one by one making sure that the expected number of satisfied equation, conditioned upon the choices made so far, never decreases. We omit the details. □

7.2.1 Earlier work

In Chapter 3, the approximation algorithm for Max Cut due to Goemans and Williamson [40] was presented and analyzed. A generalization of Max Cut is Max p-Cut, where the vertices of a graph are to be partitioned into p parts instead of two. In their approximation algorithm for Max p-Cut, Frieze and Jerrum [36] face a complication similar to ours: How to represent in a suitable way variables that can take one of p values. To do this, each vertex is represented by a vector which is one of the p vertices of a regular $(p-1)$-simplex centered at the origin in $\boldsymbol{R}^{p-1}$. If the vertices of the simplex are $\{a_1, a_2, \dots, a_p\}$, the Max p-Cut problem can be formulated as

$$\begin{array}{ll} \text{maximize} & \dfrac{p-1}{p} \displaystyle\sum_{i<i'} w_{ii'} \bigl(1 - \langle y_i, y_{i'} \rangle\bigr) \\ \text{subject to} & y_i \in \{a_1, a_2, \dots, a_p\} \text{ for all } i. \end{array}$$

The partition is formed according to $V_j = \{i \mid y_i = a_j\}$.

The natural way to relax this program to a semidefinite one is to use vectors v_i that are not constrained to the vertices of a simplex:

$$\begin{array}{ll} \text{maximize} & \dfrac{p-1}{p} \displaystyle\sum_{i<i'} w_{ii'} \bigl(1 - \langle v_i, v_{i'} \rangle\bigr) \\ \text{subject to} & \langle v_i, v_i \rangle = 1 \text{ for all } i, \\ & \langle v_i, v_{i'} \rangle \geq \frac{-1}{p-1} \text{ for all } i \neq i', \\ & v_i \in \boldsymbol{R}^n \text{ for all } i. \end{array} \tag{7.1}$$

To obtain a partition of the graph from the solution to the semidefinite program, the algorithm selects p random vectors $r_1, r_2, \dots, r_p$ uniformly distributed on the unit sphere in $\boldsymbol{R}^n$, and sets

$$V_j = \bigl\{ i \mid \langle v_i, r_j \rangle \geq \langle v_i, r_{j'} \rangle \text{ for all } j' \neq j \bigr\}.$$

This is not completely accurate as it is possible for $V_j \cap V_{j'}$ to be non-empty for $j \neq j'$, but this event has probability 0 and it is easy to avoid this problem by forming the partition in a more complicated way.

When $p = 2$ this algorithm is equivalent to the Max Cut algorithm of Goemans and Williamson.

7.2.2 Our construction

Our goal is to generalize the algorithm of Goemans and Williamson to Max 2-Lin mod p. We first construct an approximation algorithm for systems of linear equations where the equations are of the form $x_i - x_{i'} = c$.

A problem with applying the approach of Frieze and Jerrum is that it has no "metric" information, it can only express equality and non-equality. The reason for this is that the algorithm chooses the random vectors without any linear structure. Our way of achieving a linear structure is through representing each variable x_i by a constellation of p vectors, $\{u_0^i, u_1^i, \ldots, u_{p-1}^i\}$ and rounding the semidefinite solution using one random vector r. The partition would then be constructed as

$$V_j = \bigl\{x_i \mid \langle u_j^i, r\rangle \geq \langle u_{j'}^i, r\rangle \text{ for all } j' \neq j\bigr\},$$

and all variables in V_j are assigned the value $-j$. We create a consistent linear structure of these constellations by requiring that for all i, i' and all j, j', k,

$$\langle u_j^i, u_{j+k}^{i'}\rangle = \langle u_{j'}^i, u_{j'+k}^{i'}\rangle.$$

(The subscripts are interpreted mod p.) If we denote by $w_{ii'c}$ the weight of the equation $x_i - x_{i'} = c$, we can thus write our restricted version of the Max E2-Lin mod p problem as the following semidefinite program:

$$\begin{array}{ll} \text{maximize} & \displaystyle\sum_{i,i',c} w_{ii'c}\left(\frac{p-1}{p^2}\sum_{j=0}^{p-1}\langle u_j^i, u_{j+c}^{i'}\rangle + \frac{1}{p}\right) \\ \text{subject to} & \langle u_j^i, u_j^i\rangle = 1 \quad \forall i, j, \\ & \langle u_j^i, u_{j'}^i\rangle = \frac{-1}{p-1} \quad \forall i \forall j \neq j', \\ & \langle u_j^i, u_{j'}^{i'}\rangle \in \{1, \frac{-1}{p-1}\} \quad \forall i \neq i' \forall j, j', \\ & \langle u_j^i, u_{j+k}^{i'}\rangle = \langle u_{j'}^i, u_{j'+k}^{i'}\rangle \quad \forall i, i', j, j', k. \end{array} \tag{7.2}$$

To simplify the terminology, we will now define formally the constellation of vectors associated with each variable in the above program.

Definition 7.3. For each variable $x_i \in \boldsymbol{Z}_p$ we construct an object henceforth called a *simplicial porcupine* in the following way:

We take p vectors $\{u_j^i\}_{j=0}^{p-1}$ and add the following constraints to the semidefinite program:

$$\langle u_j^i, u_k^i\rangle = \begin{cases} 1 & \text{when } j = k, \\ \frac{-1}{p-1} & \text{otherwise,} \end{cases} \tag{7.3a}$$

for all i and all $j, k \in \boldsymbol{Z}_p$,

$$\langle u_j^i, u_{j+k}^{i'}\rangle = \langle u_{j'}^i, u_{j'+k}^{i'}\rangle \tag{7.3b}$$

for all i, i' and all $j, j', k \in \boldsymbol{Z}_p$, and

$$\langle u_j^i, u_k^{i'} \rangle \geq \frac{-1}{p-1} \tag{7.3c}$$

for all i, i' and all $j, k \in \boldsymbol{Z}_p$.

We can now relax the program (7.2) to a semidefinite one, and then apply the rounding procedure described above. For completeness, we write out the semidefinite relaxation:

$$\begin{array}{ll} \text{maximize} & \displaystyle\sum_{i,i',c} w_{ii'c} \left(\frac{p-1}{p^2} \sum_{j=0}^{p-1} \langle u_j^i, u_{j+c}^{i'} \rangle + \frac{1}{p} \right) \\ \text{subject to} & \langle u_j^i, u_j^i \rangle = 1 \quad \forall i, j, \\ & \langle u_j^i, u_{j'}^i \rangle = \frac{-1}{p-1} \quad \forall i \forall j \neq j', \\ & \langle u_j^i, u_{j'}^{i'} \rangle \geq \frac{-1}{p-1} \quad \forall i \neq i' \forall j, j', \\ & \langle u_j^i, u_{j+k}^{i'} \rangle = \langle u_{j'}^i, u_{j'+k}^{i'} \rangle \quad \forall i, i', j, j', k. \end{array} \tag{7.4}$$

When we are to analyze the rounding procedure, we want to study the inner products $\langle u_j^i, r \rangle$. Unfortunately, these random variables are not independent for a fixed i, and this complicates the analysis. We would obtain a simpler analysis if the vectors corresponding to a variable were orthogonal, since then the corresponding inner products would be independent. It is easy to construct such a semidefinite program. All constraints change accordingly, and for each equation the terms

$$\frac{1}{p} \sum_{j=0}^{p-1} \langle v_j^i, v_{j+c}^{i'} \rangle \tag{7.5}$$

are included in the objective function. Such a construction gives the semidefinite program

$$\begin{array}{ll} \text{maximize} & \displaystyle\sum_{i,i',c} w_{ii'c} \left(\frac{1}{p} \sum_{j=0}^{p-1} \langle v_j^i, v_{j+c}^{i'} \rangle \right) \\ \text{subject to} & \langle v_j^i, v_j^i \rangle = 1 \quad \forall i, j, \\ & \langle v_j^i, v_{j'}^i \rangle = 0 \quad \forall i \forall j \neq j', \\ & \langle v_j^i, v_{j'}^{i'} \rangle \geq 0 \quad \forall i \neq i' \forall j, j', \\ & \langle v_j^i, v_{j+k}^{i'} \rangle = \langle v_{j'}^i, v_{j'+k}^{i'} \rangle \quad \forall i, i', j, j', k. \end{array} \tag{7.6}$$

We use the same rounding procedure in both cases: x_i is assigned the value $-j$ if $\langle v_j^i, r \rangle \geq \langle v_{j'}^i, r \rangle$ for all $j' \neq j$. It is this program we will analyze in Section 7.3.

Definition 7.4. For each variable $x_i \in \boldsymbol{Z}_p$ we construct an object henceforth called an *orthogonal porcupine* in the following way:

We take p vectors $\{v^i_j\}_{j=0}^{p-1}$ and add the following constraints to the semidefinite program:

$$\langle v^i_j, v^i_k\rangle = \begin{cases} 1 & \text{when } j = k, \\ 0 & \text{otherwise,} \end{cases} \tag{7.7a}$$

for all i and all $j, k \in \boldsymbol{Z}_p$,

$$\langle v^i_j, v^{i'}_{j+k}\rangle = \langle v^i_{j'}, v^{i'}_{j'+k}\rangle \tag{7.7b}$$

for all i, i' and all $j, j', k \in \boldsymbol{Z}_p$, and

$$\langle v^i_j, v^{i'}_k\rangle \geq 0 \quad \text{for all } i, i' \text{ and all } j, k \in \boldsymbol{Z}_p. \tag{7.7c}$$

When no confusion can arise, we will simply call the above object a *porcupine*.

In fact, the simplicial and orthogonal formulations are equally good, in terms of the quality of the relaxation.

Theorem 7.5. *For Max E2-Lin mod p, the simplicial and orthogonal porcupine models achieve the same performance.*

Proof. An orthogonal porcupine $\{v^i_j\}_{j=0}^{p-1}$ can be transformed into a simplicial porcupine $\{u^i_j\}_{j=0}^{p-1}$ by letting

$$b^i = \frac{1}{p}\sum_{k=0}^{p-1} v^i_k, \tag{7.8}$$

$$u^i_j = \sqrt{\frac{p}{p-1}}\big(v^i_j - b^i\big). \tag{7.9}$$

With this transformation, the constraints (7.7b) imply the constraints (7.3b). Also, the constraints (7.7b) and (7.7c) together imply the constraints (7.3c). To see this, it is enough to show that

$$\begin{aligned} -1/p &\leq \langle v^i_j - b^i, v^{i'}_{j'} - b^{i'}\rangle \\ &= \langle v^i_j, v^{i'}_{j'}\rangle - \langle v^i_j, b^{i'}\rangle - \langle b^i, v^{i'}_{j'}\rangle + \langle b^i, b^{i'}\rangle. \end{aligned}$$

Now note that the constraints (7.7b) imply that

$$\langle v^i_j, b^{i'}\rangle = \langle b^i, v^{i'}_{j'}\rangle = \langle b^i, b^{i'}\rangle,$$

and thus

$$\langle v^i_j - b^i, v^{i'}_{j'} - b^{i'}\rangle = \langle v^i_j, v^{i'}_{j'}\rangle - \langle b^i, b^{i'}\rangle \geq -\|b^i\|\,\|b^{i'}\| = -1/p.$$

Consider the contribution to the objective function from the equation $x_i - x_{i'} = c$ in the two models. The simplicial porcupine gives

$$\begin{aligned}\frac{p-1}{p^2}\sum_{j=0}^{p-1}\langle u_j^i, u_{j+c}^{i'}\rangle + \frac{1}{p} &= \frac{1}{p}\sum_{j=0}^{p-1}\langle v_j^i, v_{j+c}^{i'}\rangle - \frac{1}{p^2}\langle\sum_{k=0}^{p-1} v_k^i, \sum_{k=0}^{p-1} v_k^{i'}\rangle + \frac{1}{p}\\ &\geq \frac{1}{p}\sum_{j=0}^{p-1}\langle v_j^i, v_{j+c}^{i'}\rangle\end{aligned}$$

(by the Cauchy-Schwarz inequality) with equality if and only if the orthogonal porcupines $\{v_j^i\}_{j=0}^{p-1}$ and $\{v_j^{i'}\}_{j=0}^{p-1}$ have the same barycenters. This can be ensured by adding the following constraints to the semidefinite program:

$$\sum_{j=0}^{p-1}\sum_{j'=0}^{p-1}\langle v_j^i, v_{j'}^{i'}\rangle = p \tag{7.10}$$

for all i, i'.

On the other hand, a simplicial porcupine $\{u_j^i\}_{j=0}^{p-1}$ can likewise be transformed into an orthogonal porcupine $\{v_j^i\}_{j=0}^{p-1}$ by letting

$$v_j^i = \sqrt{\frac{1}{p}}u_\perp + \sqrt{\frac{p-1}{p}}u_j^i, \tag{7.11}$$

where $\langle u_\perp, u_j^i\rangle = 0$ for all i, j and $\|u_\perp\| = 1$. This construction results in the barycenters of all orthogonal porcupines coinciding if the same $u_\perp$ is used for all simplicial porcupines. Also, the constraints (7.7b) will be satisfied by the orthogonal porcupine if the constraints (7.3b) are satisfied by the simplicial porcupine. This in fact implies that we even without the conditions (7.10) can assume that the barycenters of all orthogonal porcupines coincide. For, using the transformations in (7.9) and (7.11), we can transform an arbitrary family of orthogonal porcupines into a family of orthogonal porcupines with coinciding barycenters without decreasing the objective function.

The probability of the equation $x_i - x_{i'} = c$ being satisfied after the randomized rounding is

$$\begin{aligned}&p \times \Pr[x_i \leftarrow c \cap x_{i'} \leftarrow 0]\\ &= p \times \Pr\left[\bigcap_{j=0}^{p-1}\Big(\langle v_{-c}^i, r\rangle \geq \langle v_j^i, r\rangle\Big) \quad \cap \quad \bigcap_{j=0}^{p-1}\Big(\langle v_0^{i'}, r\rangle \geq \langle v_j^{i'}, r\rangle\Big)\right].\end{aligned}$$

The transformations between the different types of porcupines only involve scaling both sides of the inequalities by the same positive factor or adding the same expression to both sides. Hence $\Pr[x_i \leftarrow c \bigcap x_{i'} \leftarrow 0]$ is unaffected. □

When studying the Max E2-Lin mod p problem, we will use orthogonal porcupines. Let us show that our construction is a relaxation of Max E2-Lin mod p.

Lemma 7.6. *Given an instance of Max E2-Lin mod p with all equations of the form $x_i - x_{i'} = c$ and the corresponding semidefinite program (7.6), the optimum of the former can never be larger than the optimum of the latter.*

Proof. Suppose that we have an assignment π to the variables x_i such that x_i is assigned the value $\pi(x_i)$. Let $\{\hat{e}_j\}_{j=0}^{p-1}$ be orthonormal unit vectors in $\boldsymbol{R}^p$ and set

$$v^i_{j-\pi(x_i)} = \hat{e}_j \quad \text{for all } i \text{ and all } j \in \boldsymbol{Z}_p.$$

The sum (7.5) corresponding to an equation $x_i - x_{i'} = c$ then takes the value

$$\frac{1}{p}\sum_{j=0}^{p-1}\langle v^i_j, v^{i'}_{j+c}\rangle = \frac{1}{p}\sum_{j=0}^{p-1}\langle \hat{e}_{j+\pi(x_i)}, \hat{e}_{j+c+\pi(x_{i'})}\rangle.$$

If the equation $x_i - x_{i'} = c$ is satisfied, then $\pi(x_i) = \pi(x_{i'}) + c$, and

$$\frac{1}{p}\sum_{j=0}^{p-1}\langle \hat{e}_{j+\pi(x_i)}, \hat{e}_{j+c+\pi(x_{i'})}\rangle = 1.$$

On the other hand, if the equation is not satisfied, then $\pi(x_i) \neq \pi(x_{i'}) + c$, and

$$\frac{1}{p}\sum_{j=0}^{p-1}\langle \hat{e}_{j+\pi(x_i)}, \hat{e}_{j+c+\pi(x_{i'})}\rangle = 0.$$

Thus, the maximum of the semidefinite program can never be less than the optimum of the Max E2-Lin mod p instance. □

7.3 Our algorithms

In this section we use the relaxations constructed in Section 7.2.2 to formulate an algorithm approximating Max 2-Lin mod p within $1/(1-\kappa(p))p$, where $\kappa(p) > 0$, for all p. The algorithm is constructed in three steps. First, we describe an algorithm that works for instances of Max E2-Lin mod p where all equations are of the form $x_i - x_{i'} = c$. This algorithm is then generalized to handle instances where also equations of the form $x_i = c$ are allowed. Finally, the resulting algorithm is generalized once more to handle general Max 2-Lin mod p instances.

7.3.1 Equations of the form $x_i - x_{i'} = c$

We use the semidefinite program (7.6) constructed in Section 7.2.2. We can now formulate the algorithm to approximate Max E2-Lin mod p restricted to instances where all equations are of the form $x_i - x_{i'} = c$. Below, κ is a constant which is to be determined during the analysis of the algorithm. Given a set of linear equations, we run both Algorithm 4 and the following algorithm:

Algorithm 5. *Construct and solve the semidefinite program (7.6). Use the vectors obtained from the optimal solution to the semidefinite program to obtain an assignment to the variables x_i in the following way: A vector r is selected by independently choosing each component as an $\mathrm{N}(0,1)$ random variable. Then, for each porcupine $\{v_j^i\}_{j=0}^{p-1}$ we find the j maximizing $\langle v_j^i, r\rangle$, and set $x_i = -j$.*

We take as our result the maximum of the results obtained from Algorithms 4 and 5. By Corollary 7.2, we will always approximate the optimum within $1/(1-\kappa)p$ if the optimum weight is less than $1-\kappa$ times the weight of all equations. Thus, when analyzing the performance of Algorithm 5, we can assume that the optimum is at least $1-\kappa$ times the weight of all equations. Intuitively, this means that for most equations, the two porcupines involved will be almost perfectly aligned.

Lemma 7.7. *If the objective function is at least $1-\kappa$ times the total weight of all equations, then equations of total weight at least $1-2\kappa/\varepsilon$ times the weight of the instance have the property that the corresponding terms (7.5) in the objective function evaluate to at least $\sqrt{1-\varepsilon}$.*

Proof. Let μ be the fraction of the equations with the property that the corresponding terms (7.5) in the objective functions are less than $\sqrt{1-\varepsilon}$. Then the inequality

$$\mu\sqrt{1-\varepsilon} + (1-\mu) \geq 1-\kappa$$

must always hold. When we solve for μ we obtain $\mu \leq \kappa/(1-\sqrt{1-\varepsilon}) \leq 2\kappa/\varepsilon$. $\square$

Let us now study a fixed equation $x_i - x_{i'} = c$, where the sum of the corresponding terms in the objective function of the semidefinite program satisfies

$$\frac{1}{p}\sum_{j=0}^{p-1}\langle v_j^i, v_{j+c}^{i'}\rangle = \sqrt{1-\varepsilon}, \tag{7.12}$$

where ε is small. By the constraints in (7.7b), (7.12) implies that

$$v_{j+c}^{i'} = \sqrt{1-\varepsilon}\,v_j^i + \sqrt{\varepsilon}\,e_j^c,$$

where e_j^c is orthogonal to v_j^i and $\|e_j^c\| = 1$.

Definition 7.8. For a fixed equation $x_i - x_{i'} = c$, let $X_j = \langle v_j^i, r\rangle$, $Y_j = \langle v_{j+c}^{i'}, r\rangle$, and $Z_j = \langle e_j^c, r\rangle$.

By the construction of the porcupines and the choice of r, the X_j are independent identically distributed (from now on abbreviated i.i.d.) $\mathrm{N}(0,1)$ and the Z_j are, possibly dependent, $\mathrm{N}(0,1)$. However, for each fixed j, X_j and Z_j are independent. To show that our algorithm has a performance guarantee of at least $1/(1-\kappa)p$, it is, by Lemma 7.7, sufficient to prove the following:

Lemma 7.9. *It is possible to choose universal constants $\kappa < 1$ and $\varepsilon > 2\kappa$ such that for all primes p, and for any equation $x_i - x_{i'} = c$ whose corresponding terms (7.5) in the objective function are at least $\sqrt{1-\varepsilon}$,*

$$\Pr[\text{equation satisfied}] > \frac{1}{p(1-\kappa)(1-2\kappa/\varepsilon)}.$$

The proof of this lemma uses four lemmas about the normal distribution. Let

$$\varphi(x) = \frac{e^{-x^2/2}}{\sqrt{2\pi}}$$

and

$$\Phi(x) = \int_{-\infty}^{x} \varphi(t)\, dt.$$

If we integrate $\Phi(x)$ by parts, we obtain

$$\begin{aligned}
\sqrt{2\pi}(1-\Phi(x)) &= \int_x^\infty e^{-t^2/2}\, dt \\
&= \int_x^\infty t e^{-t^2/2} t^{-1}\, dt \\
&= \frac{e^{-x^2/2}}{x} - \int_x^\infty e^{-t^2/2} t^{-2}\, dt \\
&= \left(\frac{1}{x} - \frac{1}{x^3}\right) e^{-x^2/2} + 3\int_x^\infty e^{-t^2/2} t^{-4}\, dt.
\end{aligned}$$

The above equalities immediately imply that

$$\varphi(x)\left(\frac{1}{x} - \frac{1}{x^3}\right) < 1 - \Phi(x) < \frac{\varphi(x)}{x}, \tag{7.13}$$

when $x > 0$. This bound will be used to prove the following lemmas.

Lemma 7.10. *Let $X_0, \dots, X_{p-1}$ be independent identically distributed $\mathrm{N}(0,1)$ random variables. Denote the maximum of the X_i by $X_{(p)}$, and the second maximum by $X_{(p-1)}$. Then, for any $\delta > 0$,*

$$\begin{aligned}
&\Pr\Big[X_{(p)} \geq (1+\delta)\sqrt{2\ln p} \quad \bigcap \quad X_{(p-1)} \leq (1+\delta/2)\sqrt{2\ln p}\Big] \\
&\quad > \frac{1}{2p^{2\delta+\delta^2}(1+\delta)\sqrt{\pi \ln p}}\left(1 - \frac{1}{2\ln p} - \frac{1}{2p^{\delta}\sqrt{\pi\ln p}}\right).
\end{aligned}$$

Proof. Since the X_i are i.i.d. $\mathrm{N}(0,1)$, we know that

$$\Pr[X_{(p)} \geq x \cap X_{(p-1)} \leq y] = p\big(1-\Phi(x)\big)\Phi(y)^{p-1} \tag{7.14}$$

when $x \geq y$. We now apply the bound on $\Phi(x)$ from (7.13). This bound, together with the fact that $\delta > 0$, implies that

$$\begin{aligned} 1 - \Phi&\left((1+\delta)\sqrt{2\ln p}\right) \\ &> \frac{1}{\sqrt{2\pi}p^{(1+\delta)^2}}\left(\frac{1}{(1+\delta)\sqrt{2\ln p}} - \frac{1}{(1+\delta)^3(2\ln p)^{3/2}}\right) \\ &> \frac{1}{2p^{1+2\delta+\delta^2}(1+\delta)\sqrt{\pi\ln p}}\left(1-\frac{1}{2\ln p}\right), \end{aligned}$$

and that

$$\begin{aligned} \Phi\left((1+\delta/2)\sqrt{2\ln p}\right) &> 1 - \frac{1}{\sqrt{2\pi}p^{(1+\delta/2)^2}(1+\delta/2)\sqrt{2\ln p}} \\ &> 1 - \frac{1}{2p^{1+\delta}\sqrt{\pi\ln p}}. \end{aligned}$$

When this is inserted into (7.14), we obtain

$$\begin{aligned} \Pr&[X_{(p)} \geq x \cap X_{(p-1)} \leq y] \\ &> p\frac{1}{2p^{1+2\delta+\delta^2}(1+\delta)\sqrt{\pi\ln p}}\left(1-\frac{1}{2\ln p}\right)\left(1-\frac{1}{2p^{1+\delta}\sqrt{\pi\ln p}}\right)^{p-1} \\ &> p\frac{1}{2p^{1+2\delta+\delta^2}(1+\delta)\sqrt{\pi\ln p}}\left(1-\frac{1}{2\ln p}\right)\left(1-\frac{1}{2p^{1+\delta}\sqrt{\pi\ln p}}\right)^{p}. \end{aligned} \tag{7.15}$$

Consider the last factor in (7.15). It is of the form $f(x) = (1-x)^p$, and this function is convex on the interval $(-\infty, 1]$. By convexity, $f(x) \geq f(0) + f'(0)x = 1 - px$. Apply this relation to the last factor in (7.15):

$$\left(1-\frac{1}{2p^{1+\delta}\sqrt{\pi\ln p}}\right)^p > 1 - p\frac{1}{2p^{1+\delta}\sqrt{\pi\ln p}} = 1 - \frac{1}{2p^{\delta}\sqrt{\pi\ln p}}.$$

Hence

$$\begin{aligned} \left(1-\frac{1}{2\ln p}\right)&\left(1-\frac{1}{2p^{1+\delta}\sqrt{\pi\ln p}}\right)^p \\ &> \left(1-\frac{1}{2\ln p}\right)\left(1-\frac{1}{2p^{\delta}\sqrt{\pi\ln p}}\right) \\ &= 1 - \frac{1}{2\ln p} - \frac{1}{2p^{\delta}\sqrt{\pi\ln p}} + \frac{1}{4p^{\delta}\sqrt{\pi\ln p}\ln p} \\ &> 1 - \frac{1}{2\ln p} - \frac{1}{2p^{\delta}\sqrt{\pi\ln p}}. \end{aligned}$$

Insert this into (7.15) and the lemma follows. ∎

Lemma 7.11. *Let X and Z be i.i.d.* $\mathrm{N}(0,1)$ *and* $\varepsilon \in [0,1]$. *Then, for any* $\delta > 0$,

$$\Pr\left[\left|\left(1-\sqrt{1-\varepsilon}\right)X - \sqrt{\varepsilon}Z\right| > \frac{\delta}{4}\sqrt{2(1-\varepsilon)\ln p}\right] \\ \leq \frac{4p^{-\delta^2(1-\varepsilon)/32\varepsilon}}{\delta}\sqrt{\frac{2\varepsilon}{(1-\varepsilon)\pi\ln p}}.$$

Proof. Let $W = (1-\sqrt{1-\varepsilon})X - \sqrt{\varepsilon}Z$. Since X and Z are independent, $W \in \mathrm{N}(0,\sigma)$, where

$$\sigma = \sqrt{\left(1-\sqrt{1-\varepsilon}\right)^2 + \varepsilon} \leq \sqrt{2\varepsilon}.$$

Since $\Pr[|W| > w] = 2(1-\Phi(w/\sigma))$, we can use (7.13).

$$\begin{aligned}\Pr\left[|W| > \frac{\delta}{4}\sqrt{2(1-\varepsilon)\ln p}\right] &= 2\left(1-\Phi\left(\frac{\delta}{4\sigma}\sqrt{2(1-\varepsilon)\ln p}\right)\right) \\ &< 2\frac{4\sigma}{\delta\sqrt{2(1-\varepsilon)\ln p}} \times \frac{p^{-\delta^2(1-\varepsilon)/16\sigma^2}}{\sqrt{2\pi}} \\ &\leq \frac{4p^{-\delta^2(1-\varepsilon)/32\varepsilon}}{\delta}\sqrt{\frac{2\varepsilon}{(1-\varepsilon)\pi\ln p}}.\end{aligned}$$

□

Lemma 7.12. *Let $X_0, \dots, X_{p-1}$ be i.i.d.* $\mathrm{N}(0,1)$ *random variables. Denote the maximum of the X_i by $X_{(p)}$, and the second maximum by $X_{(p-1)}$. Then*

$$\Pr\left[X_{(p)} > X_{(p-1)} + \delta\right] > 1 - \frac{p^2\delta}{(p-1)\sqrt{2\pi}}.$$

Proof. Since the X_i are independent,

$$\Pr\left[X_{(p)} > X_{(p-1)} + \delta\right] = p \times \Pr\left[\bigcap_{i=1}^{p-1} X_0 > X_i + \delta\right].$$

To compute the latter probability we condition on X_0.

$$\Pr\left[\bigcap_{i=1}^{p-1} X_0 > X_i + \delta\right] = \int_{-\infty}^{\infty} \Phi^{p-1}(x-\delta)\varphi(x)\,dx.$$

To bound $\Phi^{p-1}(x-\delta)$, we use the mean value theorem. (In the following equations, $\xi \in [x-\delta, x]$.)

$$\begin{aligned}
\Phi^{p-1}(x-\delta) &= \big(\Phi(x) - \delta\varphi(\xi)\big)^{p-1} \\
&\geq \Phi^{p-1}(x) - p\delta\varphi(\xi)\Phi^{p-2}(x) \\
&\geq \Phi^{p-1}(x) - p\delta\Phi^{p-2}(x) \max_{y\in(-\infty,\infty)} \varphi(y) \\
&= \Phi^{p-1}(x) - \frac{p\delta}{\sqrt{2\pi}}\Phi^{p-2}(x).
\end{aligned}$$

From this bound on $\varphi(x)$, we obtain

$$\begin{aligned}
&\int_{-\infty}^{\infty} \Phi^{p-1}(x-\delta)\varphi(x)\,dx \\
&\geq \int_{-\infty}^{\infty} \Phi^{p-1}(x)\varphi(x)\,dx - \frac{p\delta}{\sqrt{2\pi}}\int_{-\infty}^{\infty} \Phi^{p-2}(x)\varphi(x)\,dx \\
&= \frac{1}{p} - \frac{p\delta}{(p-1)\sqrt{2\pi}}
\end{aligned}$$

and the lemma follows. □

Lemma 7.13. *Let X and Z be i.i.d.* $\mathrm{N}(0,1)$ *and* $\varepsilon \in [0,1]$. *Then, for any* $\delta > 0$,

$$\Pr\Big[\Big|\Big(1-\sqrt{1-\varepsilon}\Big)X - \sqrt{\varepsilon}Z\Big| > \delta/2\Big] \leq \frac{4}{\delta}\sqrt{\frac{\varepsilon}{\pi}}.$$

Proof. Since X and Z are independent,

$$\Big(1-\sqrt{1-\varepsilon}\Big)X - \sqrt{\varepsilon}Z \in \mathrm{N}(0,\sigma),$$

where

$$\sigma = \sqrt{\Big(1-\sqrt{1-\varepsilon}\Big)^2 + \varepsilon} \leq \sqrt{2\varepsilon}.$$

Thus,

$$\begin{aligned}
\Pr\Big[\Big|\Big(1-\sqrt{1-\varepsilon}\Big)X - \sqrt{\varepsilon}Z\Big| > \delta/2\Big] &\leq 2\Big(1-\Phi\big(\delta/2\sigma\big)\Big) \\
&\leq \frac{4\sigma}{\delta\sqrt{2\pi}} \\
&\leq \frac{4}{\delta}\sqrt{\frac{\varepsilon}{\pi}}.
\end{aligned}$$

□

Proof of Lemma 7.9. The randomized rounding succeeds if the "chosen" vectors are v_j^i and $v_{j+c}^{i'}$, respectively, for some j. Another way to state this is that we want to estimate the probability that some j maximizes Y_j, given that the very same j maximizes X_j.

We will first show that the theorem holds for large p: Let $A(\delta)$ be the event that the largest X_j is at least $(1+\delta)\sqrt{2\ln p}$ and that all other X_j are at most $(1+\delta/2)\sqrt{2\ln p}$. By Lemma 7.10,

$$\Pr[A(\delta)] > \frac{1}{2p^{2\delta+\delta^2}(1+\delta)\sqrt{\pi\ln p}}\left(1-\frac{1}{2\ln p}-\frac{1}{2p^{\delta}\sqrt{\pi\ln p}}\right).$$

Next, let us study the Z_j. Let

$$B(\delta,\varepsilon) = \bigcap_{j=0}^{p-1}\left\{|X_j - Y_j| < \frac{\delta}{4}\sqrt{(1-\varepsilon)2\ln p}\right\}.$$

Since $Y_j = \sqrt{1-\varepsilon}X_j + \sqrt{\varepsilon}Z_j$, we can use Lemma 7.11 to obtain

$$\Pr[\overline{B(\delta,\varepsilon)}] < \frac{4p^{1-\delta^2(1-\varepsilon)/32\varepsilon}}{\delta}\sqrt{\frac{2\varepsilon}{(1-\varepsilon)\pi\ln p}}.$$

The equation is satisfied if both $A(\delta)$ and $B(\delta,\varepsilon)$ occur. The probability that this happens can be bounded by

$$\Pr[A(\delta)\cap B(\delta,\varepsilon)] \geq \Pr[A(\delta)] - \Pr[\overline{B(\delta,\varepsilon)}].$$

It is now clear that we can choose constants δ, ε and κ that give the probability we want for sufficiently large p. For instance, $\delta = 10^{-2}$, $\varepsilon = 10^{-7}$ and $\kappa = 10^{-8}$ work when $p \geq 13$.

Now it remains to be shown that the claim is valid also for small p. Let $C(\delta)$ be the event that the difference between the largest and the second largest X_j is at least δ, and let $D(\delta)$ be the event that, for all j, $|X_j - Y_j| \leq \delta/2$. By Lemmas 7.12 and 7.13,

$$\Pr[C(\delta)] \geq 1 - \frac{p^2\delta}{(p-1)\sqrt{2\pi}},$$

$$\Pr[\overline{D(\delta)}] \leq \frac{4p}{\delta}\sqrt{\frac{\varepsilon}{\pi}}.$$

The equation is satisfied if both $C(\delta)$ and $D(\delta)$ occur. The probability that this happens can be bounded by

$$\Pr[C(\delta)\cap D(\delta)] \geq \Pr[C(\delta)] - \Pr[\overline{D(\delta)}],$$

and a simple calculation shows that $\delta = 10^{-2}$, $\varepsilon = 10^{-7}$ and $\kappa = 10^{-8}$ work when $p \leq 11$. $\square$

Putting the pieces together we obtain:

Theorem 7.14. *There exists a universal constant κ such that there exists, for all primes p, a randomized polynomial time algorithm approximating systems of linear equations mod p of the form $x_i - x_{i'} = c$ within $1/(1-\kappa)p$.*

Proof. The algorithm is as described above. Denote by w the total weight of the instance. If the optimum is at most $(1-\kappa)w$, Algorithm 4 approximates the solution within $1/(1-\kappa)p$.

Otherwise, by Lemma 7.7, equations with total weight at least $(1-2\kappa/\varepsilon)w$ have the property that the corresponding terms in the objective function in the semidefinite program evaluate to at least $\sqrt{1-\varepsilon}$ in the optimal solution. By Lemma 7.9, if we choose $\varepsilon = 10^{-7}$ and $\kappa = 10^{-8}$, these equations are satisfied with probability at least $1/p(1-\kappa)(1-2\kappa/\varepsilon)$, over the choice of the random vector r. Thus, the expected weight of the solution obtained by the rounding is at least $w/p(1-\kappa)$. □

It is straightforward to adapt the algorithm to handle equations with one unknown: Simply introduce a new variable x_0 which should take the value zero. Each equation of the form $x_i = c$ is replaced by $x_i - x_0 = c$. If $x_0 \neq 0$ in the optimal solution, we transform the solution according to $x_i \leftarrow x_i - x_0$. This new assignment to the variables satisfies exactly the same equations as the original one.

The bound in Theorem 7.14 is far from tight. For $p = 3$, in practice probably the most interesting value of p, the performance guarantee was evaluated numerically and was found to be 0.78. This can be compared with Frieze and Jerrum's 0.80-approximation algorithm for the special case Max 3-Cut [36]

Finally, since nothing in Section 7.3.1 actually uses that p is prime, the results hold also for composite p.

7.3.2 General equations

In this section we extend the algorithm from Section 7.3.1 to handle general Max 2-Lin mod p instances. We do this by associating $p-1$ porcupines, $\{v_j^{i,1}\}_{j=0}^{p-1}$ up to $\{v_j^{i,p-1}\}_{j=0}^{p-1}$, with each variable x_i. These porcupines are supposed to represent x_i, $2x_i$, up to $(p-1)x_i$, respectively. The porcupines are constructed as described in Definition 7.4, with (7.7) generalized to

$$\langle v_j^{i,\ell}, v_k^{i,\ell} \rangle = \begin{cases} 1 & \text{when } j = k, \\ 0 & \text{otherwise,} \end{cases} \tag{7.16a}$$

for all i, all $j, k \in \boldsymbol{Z}_p$, and all $\ell \in \boldsymbol{Z}_p^*$;

$$\langle v_j^{i,\ell}, v_{j+k}^{i',\ell'} \rangle = \langle v_{j'}^{i,\ell}, v_{j'+k}^{i',\ell'} \rangle \tag{7.16b}$$

for all i, i', all $j, j', k \in \boldsymbol{Z}_p$, and all $\ell, \ell' \in \boldsymbol{Z}_p^*$;

$$\sum_{j=0}^{p-1} \langle v_j^{i,\ell}, v_{j''}^{i'',\ell''} \rangle = \sum_{j=0}^{p-1} \langle v_j^{i',\ell'}, v_{j''}^{i'',\ell''} \rangle \tag{7.16c}$$

for all i, i', i'', all $j'' \in \boldsymbol{Z}_p$ and all $\ell, \ell', \ell'' \in \boldsymbol{Z}_p^*$; and

$$\langle v_j^{i,\ell}, v_k^{i',\ell'} \rangle \geq 0 \tag{7.16d}$$

for all i, i', all $j, k \in \boldsymbol{Z}_p$, and all $\ell, \ell' \in \boldsymbol{Z}_p^*$.

We would want the porcupines to be dependent in such a way that $x_i = c$ is equivalent to $kx_i = kc$, but since the resulting constraint is not linear, this seems hard to achieve. Instead, we allow the porcupines corresponding to the same variable to vary freely. Somewhat surprisingly, it turns out that this enables us to construct an algorithm that approximates Max 2-Lin mod p within $1/(p - \kappa(p))$, where $\kappa(p) > 0$ for all p but goes towards zero as p grows towards infinity.

To handle equations of the form $ax_i = c$ we introduce a new variable x_0. Our algorithm will be designed in such a way that x_0 always gets the value 0. Each equation $ax_i = c$ can thus be changed to $ax_i - x_0 = c$. For each equation $ax_i - bx_{i'} = c$ we include the terms

$$\frac{1}{p(p-1)} \sum_{k=1}^{p-1} \sum_{j=0}^{p-1} \langle v_j^{i,ka}, v_{j+kc}^{i',kb} \rangle \tag{7.17}$$

in the objective function.

Lemma 7.15. *Given an instance of Max 2-Lin mod p and the corresponding semidefinite program constructed as described above, the optimum of the former can never be larger than the optimum of the latter.*

Proof. Suppose that we have an assignment π to the variables x_i. Let $\{\hat{e}_j\}_{j=0}^{p-1}$ be orthonormal unit vectors in $\boldsymbol{R}^p$ and set

$$v_{j-\ell\pi(x_i)}^{i,\ell} = \hat{e}_j$$

for all i, all $j \in \boldsymbol{Z}_p$ and all $\ell \in \boldsymbol{Z}_p^*$. The terms (7.17) corresponding to an equation $ax_i - bx_{i'} = c$ are then

$$\begin{aligned} &\frac{1}{p(p-1)} \sum_{k=1}^{p-1} \sum_{j=0}^{p-1} \langle v_j^{i,ka}, v_{j+kc}^{i',kb} \rangle \\ &\quad = \frac{1}{p(p-1)} \sum_{k=1}^{p-1} \sum_{j=0}^{p-1} \langle \hat{e}_{j+ka\pi(x_i)}, \hat{e}_{j+kc+kb\pi(x_{i'})} \rangle. \end{aligned}$$

If the equation is satisfied by the assignment π, then $ka\pi(x_i) = kb\pi(x_{i'}) + kc$, and

$$\frac{1}{p(p-1)} \sum_{k=1}^{p-1} \sum_{j=0}^{p-1} \langle \hat{e}_{j+ka\pi(x_i)}, \hat{e}_{j+kc+kb\pi(x_{i'})} \rangle = 1.$$

On the other hand, if the equation is not satisfied, then $ka\pi(x_i) \neq kb\pi(x_{i'}) + kc$, and

$$\frac{1}{p(p-1)} \sum_{k=1}^{p-1} \sum_{j=0}^{p-1} \langle \hat{e}_{j+ka\pi(x_i)}, \hat{e}_{j+kc+kb\pi(x_{i'})} \rangle = 0.$$

Thus, the maximum of the semidefinite program can never be less than the optimum of the Max 2-Lin mod p instance. □

Below, $\kappa(p)$ is some function, which is to be determined during the analysis of the algorithm. We construct an approximation algorithm for Max 2-Lin mod p by generalizing Algorithm 5 as follows:

Algorithm 6. *Construct and solve the above semidefinite program. Use the vectors obtained from the optimal solution to the semidefinite program to obtain an assignment to the variables x_i in the following way: A vector r is selected by independently choosing each component as an* $\mathrm{N}(0, 1)$ *random variable. Then we do the following:*

Find the $j \in \boldsymbol{Z}_p$ maximizing $\langle v_j^{0,1}, r \rangle$.
Set $t \leftarrow j$.
For each $i \in [0..n]$,
  For all $j \in \boldsymbol{Z}_p$, set $q_{i,j} \leftarrow 0$.
  For all $k \in \boldsymbol{Z}_p^*$,
    Find the $j \in \boldsymbol{Z}_p$ maximizing $\langle v_j^{i,k}, r \rangle$.
    Set $q_{i,k^{-1}(j-t)} \leftarrow 1$.
  Set $Q_i \leftarrow \sum_k q_{i,k}$.
  For all $j \in \boldsymbol{Z}_p$, set $q_{i,j} \leftarrow q_{i,j}/Q_i$.

Finally, given the resulting $q_{i,j}$, each variable x_i, except for x_0, is given the value $-j$ with probability $q_{i,j}$. The variable x_0 is given the value 0.

Remark 7.16. By the choice of t in Algorithm 6 above, $q_{0,0}$ will always be non-zero. This turns out to be essential in the analysis.

To obtain our estimate of the optimum of the Max 2-Lin mod p instance, we take the maximum of the results obtained from Algorithms 4 and 6.

By Corollary 7.2 and Lemma 7.15, Algorithm 4 is a $1/(1-\kappa)p$-approximation algorithm if the optimum weight of the semidefinite program is less than $1-\kappa$ times the weight of all equations. Thus, when analyzing the performance of Algorithm 6, we can assume that the optimum of the semidefinite program is at least $1-\kappa$ times the weight of all equations. By this assumption and Lemma 7.7, equations of total weight at least $1-2\kappa/\varepsilon$ times the weight of the instance have the property that the sum of the corresponding terms (7.17) in the objective functions is at least $\sqrt{1-\varepsilon}$. Let us now study an arbitrary equation $ax_i - bx_{i'} = c$ with that property. I.e.,

$$\frac{1}{p(p-1)}\sum_{k=1}^{p-1}\sum_{j=0}^{p-1}\langle v_j^{i,ka}, v_{j+kc}^{i',kb}\rangle \geq \sqrt{1-\varepsilon}. \tag{7.18}$$

We want to show that this equation is satisfied with probability slightly larger than $1/p$. Let us study the details of the selection procedure in Algorithm 6. Informally, we expect the following:

- By the condition in (7.18) we expect the vectors $v_j^{i,ka}$ and $v_{j+kc}^{i',kb}$ to be almost perfectly aligned, for all j and k.
- For each k, this should imply that if some j maximizes $\langle v_j^{i,ka}, r\rangle$ then, with high probability over the choice of r, we will have that $j' = j + kc$ maximizes $\langle v_{j'}^{i',kb}, r\rangle$.
- In terms of $q_{i,j}$ this means that the equivalence

  $$q_{i,a^{-1}j} \neq 0 \iff q_{i',b^{-1}(j+c)} \neq 0$$

  should hold for each j with high probability.
- If the above equivalence holds for all j, the randomized assignment at the end of Algorithm 6 will find a satisfying assignment with probability greater than $1/p$.

We now formalize this intuition.

Definition 7.17. Let $X_j^k = \langle v_j^{i,ak}, r\rangle$ and $Y_j^k = \langle v_{j+kc}^{i',bk}, r\rangle$.

Remark 7.18. By the construction of the porcupines and the choice of r, the X_j^k are i.i.d. $\mathrm{N}(0,1)$.

Definition 7.19. Let ε_k be defined by the relation

$$\frac{1}{p}\sum_{j=0}^{p-1}\langle v_j^{i,ka}, v_{j+kc}^{i',kb}\rangle = \sqrt{1-\varepsilon_k}. \tag{7.19}$$

Remark 7.20. By the constraints in (7.16), the above definitions imply that

$$Y_j^k = \sqrt{1-\varepsilon_k}X_j^k + \sqrt{\varepsilon_k}Z_j^k, \tag{7.20}$$

where $Z_j^k \in \mathrm{N}(0,1)$. Furthermore, X_j^k and Z_j^k are independent.

Lemma 7.21. *Let $ax_i - bx_{i'} = c$ be an arbitrary equation with the property that the corresponding terms in the objective function satisfy (7.18), and let A_k be the event that the same j maximizes X_j^k and Y_j^k. Then, for all $\delta > 0$,*

$$\Pr\left[\overline{A_k}\right] \leq \frac{p^2\delta}{(p-1)\sqrt{2\pi}} + \frac{4p}{\delta}\sqrt{\frac{\varepsilon_k}{\pi}}.$$

Proof. Let $X_{(p)}^k$ and $X_{(p-1)}^k$ be the maximum and the second maximum, respectively, of the X_j^k. Define the events $B_k(\delta)$ and $C_k(\delta)$ as follows:

$$B_k(\delta) = \left\{X_{(p)}^k > X_{(p-1)}^k + \delta\right\},$$
$$C_k(\delta) = \bigcap_{i=0}^{p-1} \left\{\left|X_i^k - Y_i^k\right| < \frac{\delta}{2}\right\}.$$

If both $B_k(\delta)$ and $C_k(\delta)$ occur, then A_k must occur. Furthermore, if there exists some δ such that $B_k(\delta)$ and $C_k(\delta)$ both occur with high probability, A_k will also occur with high probability. For,

$$B_k(\delta) \cap C_k(\delta) \subseteq A_k \implies \Pr\left[\overline{A_k}\right] \leq \Pr\left[\overline{B_k(\delta)}\right] + \Pr\left[\overline{C_k(\delta)}\right]. \tag{7.21}$$

By Lemma 7.12 we obtain the bound

$$\Pr\left[\overline{B_k(\delta)}\right] \leq \frac{p^2\delta}{(p-1)\sqrt{2\pi}}, \tag{7.22}$$

and by (7.20) and Lemma 7.13 we obtain

$$\Pr\left[\overline{C_k(\delta)}\right] \leq \frac{4p}{\delta}\sqrt{\frac{\varepsilon_k}{\pi}}. \tag{7.23}$$

When (7.21), (7.22) and (7.23) are combined, the proof follows. □

Lemma 7.22. *For fixed i and i', let A_k be the event that the same j maximizes X_j^k and Y_j^k. Then, if A_k occur for all k, we are ensured that $q_{i,j} = q_{i',b^{-1}(aj+c)}$ for all $j \in \boldsymbol{Z}_p$.*

Proof. Fix i and i'. Initially in the algorithm, all $q_{i,j}$ are zero. By the construction of $q_{i,j}$ in the algorithm, the fact that A_k occur for all k implies that

$$q_{i,a^{-1}k^{-1}(j'-t)} \neq 0 \iff q_{i',b^{-1}(k^{-1}(j'-t)+c)} \neq 0$$

for all j' and k. If we substitute $j \leftarrow k^{-1}(j'-t)$, we obtain that $q_{i,a^{-1}j}$ is non-zero if and only if $q_{i',b^{-1}(j+c)}$ is non-zero. But since

$$\sum_{j=0}^{p-1} q_{i,a^{-1}j} = \sum_{j=0}^{p-1} q_{i,j} = 1,$$
$$\sum_{j=0}^{p-1} q_{i',b^{-1}(j+c)} = \sum_{j=0}^{p-1} q_{i',j} = 1,$$

this implies that $q_{i,j} = q_{i',b^{-1}(aj+c)}$ for all $j \in \boldsymbol{Z}_p$. □

Lemma 7.23. *Let $ax_i - bx_{i'} = c$ be an arbitrary equation with the property that the corresponding terms in the objective function satisfy (7.18). Then,*

$$\Pr\left[\bigcap_{j\in \boldsymbol{Z}_p} q_{i,j} = q_{i',b^{-1}(aj+c)}\right] \geq 1 - \frac{p^2\delta}{\sqrt{2\pi}} - \frac{4p(p-1)}{\delta}\sqrt{\frac{\varepsilon}{\pi}},$$

where $\delta > 0$ is arbitrary.

Proof. By Lemmas 7.21 and 7.22,

$$\begin{aligned}\Pr\left[\bigcap_{j=0}^{p-1} q_{i,a^{-1}j} = q_{i',b^{-1}(j+c)}\right] &\geq \Pr\left[\bigcap_{k=1}^{p-1} A_k\right] \\ &\geq 1 - \sum_{k=1}^{p-1} \Pr\left[\overline{A_k}\right] \\ &\geq 1 - \frac{p^2\delta}{\sqrt{2\pi}} - \frac{4p}{\delta\sqrt{\pi}}\sum_{k=1}^{p-1}\sqrt{\varepsilon_k}\,.\end{aligned} \tag{7.24}$$

Since the function $x \mapsto \sqrt{1-x}$ is concave when $x \in [0,1]$, we can apply Jensen's inequality to show that

$$\sqrt{1-\varepsilon} \leq \sum_{k=1}^{p-1} \frac{\sqrt{1-\varepsilon_k}}{p-1} \leq \sqrt{1 - \sum_{k=1}^{p-1}\frac{\varepsilon_k}{p-1}},$$

where the first inequality follows from (7.18) and (7.19), and the second from Jensen's inequality. Thus,

$$\sum_{k=1}^{p-1} \frac{\varepsilon_k}{p-1} \leq \varepsilon. \tag{7.25}$$

Using the Cauchy-Schwartz inequality, we obtain from (7.25) the bound

$$\sum_{k=1}^{p-1} \sqrt{\varepsilon_k} \le \sqrt{(p-1)\sum_{k=1}^{p-1} \varepsilon_k} \le (p-1)\sqrt{\varepsilon}.$$

When this is inserted into (7.24), the proof follows. □

Lemma 7.24. *If $q_{0,0} > 0$ and $q_{i,j} = q_{i',b^{-1}(aj+c)}$ for all i, i' and all $j \in \boldsymbol{Z}_p$, then the equation $ax_i - bx_{i'} = c$ will be satisfied with probability at least $1/(p-1)$.*

Proof. By the construction of the system of linear equations there are no equations $ax_i - bx_{i'} = c$ where $i = 0$. If $i' \neq 0$ the $q_{i,j}$ and $q_{i',j}$, computed using the probabilistic construction described above, are used to independently assign values to x_i and $x_{i'}$. Thus,

$$\Pr[\text{equation satisfied}] = \sum_j q_{i,j} q_{i',b^{-1}(aj+c)} = \sum_j q_{i,j}^2,$$

where the second equality follows from the initial requirement in the formulation of the lemma. By the construction of Algorithm 6, all $q_{i,j}$ can, for each fixed i, assume only two values, one of which is zero. The other value $q_{i,j}$ can assume is $1/m$, for some $m \in [1, p-1]$. This implies that

$$\sum_j q_{i,j}^2 = m \times \frac{1}{m^2} \ge \frac{1}{p-1},$$

since exactly m of the $q_{i,j}$ assume the value $1/m$.

If $i' = 0$ we know that $b = 1$ and $x_{i'} = 0$. Then

$$\Pr[\text{equation satisfied}] = q_{i,-a^{-1}c} = q_{0,0}.$$

Since $q_{0,0} \neq 0$ we know, by the construction of Algorithm 6, that $q_{0,0} \ge 1/(p-1)$, and the proof follows. □

Theorem 7.25. *It is possible to choose $\kappa(p) > 0$ and $\varepsilon(p) > 0$ such that, for all primes p,*

$$\Pr[\text{equation satisfied}] > \frac{1}{p(1-\kappa)(1-2\kappa/\varepsilon)}$$

for all equations with the property that the corresponding terms in the objective function are at least $\sqrt{1-\varepsilon}$.

Proof. To prove the theorem, it suffices to show that

$$p(1-\kappa)(1-2\kappa/\varepsilon)\,\Pr[\text{equation satisfied}] > 1.$$

It follows immediately from the construction of Algorithm 6, together with Lemmas 7.21–7.24, that

$$\Pr[\text{equation satisfied}] > \frac{1}{p-1}\left(1 - \frac{p^2\delta}{\sqrt{2\pi}} - \frac{4p(p-1)}{\delta}\sqrt{\frac{\varepsilon}{\pi}}\right) \tag{7.26}$$

for all equations where the sum of the corresponding terms in the objective function is at least $\sqrt{1-\varepsilon}$. As an ansatz, we choose

$$\begin{aligned}
\delta(p) &= \frac{c_1\sqrt{2\pi}}{p^3}, \\
\varepsilon(p) &= \frac{c_1^2 c_2^2 \pi^2}{8p^{10}(p-1)^2}, \\
\kappa(p) &= \frac{c_1^2 c_2^2 c_3 \pi^2}{16p^{11}(p-1)^2},
\end{aligned}$$

for some positive constants c_1, c_2 and c_3. When we use this ansatz in (7.26) we obtain

$$\Pr[\text{equation satisfied}] > \frac{1}{p}\left(1 + \frac{1-c_1-c_2}{p} - \frac{c_1+c_2}{p^2}\right).$$

Thus,

$$\begin{aligned}
&p(1-\kappa)(1-2\kappa/\varepsilon) \times \Pr[\text{equation satisfied}] \\
&\quad > \left(1 - \frac{c_3}{p} - \frac{c_1^2c_2^2c_3\pi^2}{16p^{11}(p-1)^2}\left(1-\frac{c_3}{p}\right)\right)\left(1 + \frac{1-c_1-c_2}{p} - \frac{c_1+c_2}{p^2}\right) \\
&\quad > \left(1 + \frac{1-c_1-c_2-c_3}{p} - \frac{c_1+c_2+c_3}{p^2} - \left(1+\frac{1}{p}\right)\frac{c_1^2c_2^2c_3\pi^2}{16p^{11}(p-1)^2}\right).
\end{aligned}$$

From this, it is clear that it is possible to obtain

$$\Pr[\text{equation satisfied}] > \frac{1}{p(1-\kappa)(1-2\kappa/\varepsilon)}$$

for all primes p by choosing $c_1 = c_2 = c_3 = 1/5$. □

As an immediate corollary, the main theorem follows. It is proved in exactly the same way as Theorem 7.14.

Theorem 7.26. *For all primes p, there exists a randomized polynomial time algorithm approximating Max 2-Lin mod p within $1/(1-\kappa(p))p$, where $\kappa(p) > 0$ for all p.*

Proof. The algorithm is as described above. Denote by w the total weight of the instance. If the optimum is at most $(1-\kappa)w$, Algorithm 4 approximates the solution within $1/(1-\kappa)p$.

Otherwise, Lemma 7.7 tells us that equations with total weight at least $(1 - 2\kappa(p)/\varepsilon(p))w$ have the property that the corresponding terms in the objective function in the semidefinite program evaluate to at least $\sqrt{1-\varepsilon(p)}$ in the optimal solution. By Theorem 7.25, there exists $\kappa(p) > 0$ and $\varepsilon(p) > 0$ such that these equations are satisfied with probability at least $1/p(1-\kappa(p))(1-2\kappa(p)/\varepsilon(p))$, over the choice of the random vector r. Thus, the expected weight of the solution obtained by the rounding is at least $w/p(1-\kappa(p))$. □

If we use the explicit value of $\kappa(p)$ from the proof of Theorem 7.25, we see that Max 2-Lin mod p is approximable within $1/(p-\Theta(p^{-12}))$. It is presumably possible to improve the bound significantly by making a more careful analysis.

It is possible to generalize the algorithm to Max 2-Lin mod m for composite m: First notice that since equations where $\gcd(a,b,m)$ does not divide c can never be satisfied, we can remove them from the instance. Assume that the total weight of all remaining equations is w. If the optimum is less than $(1-\kappa)w$, there is nothing to prove since we can simply apply Algorithm 4, while if it is at least $(1-\kappa)w$ we consider a prime factor p of m and proceed as follows. We determine values $\{d_i\}_{i=1}^n$ mod p such that when setting $x_i = d_i + py_i$ we get a system mod m/p in y_i such that the weight of the satisfiable equations is at least $w/p(1-\kappa(p))$. The result then follows by applying Algorithm 4 to this system, resulting in a solution that satisfies equations of weight at least $w/m(1-\kappa(p))$. The condition that an equation remains satisfiable can be formulated as a linear equation mod p: Consider the equation $ax_i - bx_{i'} = c \bmod m$. With the transformation $x_i = d_i + py_i$ we get $ad_i - bd_{i'} + p(ay_i - by_{i'}) = c \bmod m$, and this equation is satisfiable if and only if $ad_i - bd_{i'} = c \bmod p$. By the assumption that it is possible to find a solution mod m that satisfies almost all equations, desired values d_i can be found by the approximation algorithm for a prime modulus.

7.4 Max p-Cut and comparison to the algorithm of Frieze and Jerrum

In this section, we go back to simplicial porcupines to simplify the comparison with the algorithm of Frieze and Jerrum [36], which is described in Section 7.2.1. We observe that Max p-Cut is a special case of Max E2-Lin mod p: That the edge

(i, i') is to be cut is equivalent to exactly one of the equations $x_i - x_{i'} = c$, for $c = 1, 2, \ldots, p-1$, being satisfied. This corresponds to the term

$$\sum_{c=1}^{p-1}\left(\frac{p-1}{p^2}\sum_{j=0}^{p-1}\langle u_j^i, u_{j+c}^{i'}\rangle + \frac{1}{p}\right)$$

in the objective function. Note that if we use the fact that $\sum_j u_j^i = 0$ for all i, we obtain exactly the same objective function as Frieze and Jerrum used. Thus, it is possible to solve Max p-Cut by formulating the instance as a Max E2-Lin mod p instance and solving it using the methods developed in Section 7.3.1. This may produce a result closer to the optimum.

Another, seemingly good, strategy to improve the algorithm of Frieze and Jerrum is to change the rounding procedure by adding constraints forcing the random vectors to be far apart.

We show that the two approaches outlined above to some extent are equivalent to the relaxation (7.1) with the original randomized rounding strategy. Notice, however, that Frieze and Jerrum's semidefinite program cannot be used for Max E2-Lin mod p as their objective function cannot represent equations of the form $x_i - x_{i'} = c$.

7.4.1 A new rounding scheme

Frieze and Jerrum round the solution to their semidefinite program using p random vectors $r_0, \ldots, r_{p-1}$ where the components of each r_i can be chosen as independent $\mathrm{N}(0, 1/\sqrt{n})$ variables. At first, it seems that it would be better to instead choose a random porcupine.

Definition 7.27. A *random orthogonal porcupine* is a porcupine chosen as follows: The first vector s_0 in the porcupine is chosen uniformly at random. Then, for each $i \geq 1$, the vector s_i is chosen uniformly at random from the subspace orthogonal to the space spanned by the vectors $s_0, \ldots, s_{i-1}$. Finally all vectors are normalized. When no confusion can arise, we will simply call the above object a *random porcupine*.

One could also imagine using a *random simplicial porcupine*, defined in the obvious way. We note in passing that a theorem analogous to Theorem 7.5 holds for random porcupines.

Theorem 7.28. *Rounding using a random orthogonal porcupine is equivalent to rounding using a random simplicial porcupine.*

Proof. Let $\{s_i\}_{i=0}^{p-1}$ be a random orthogonal porcupine and

$$s'_i = \sqrt{\frac{p}{p-1}}\left(s_i - \frac{1}{p}\sum_j s_j\right).$$

It is easy to verify that $\{s'_i\}_{i=0}^{p-1}$ is a random simplicial porcupine. The probability that the edge (i, i') is not cut after the rounding is

$$p \times \Pr\left[\bigcap_{j=1}^{p-1}\Big(\langle v^i, s'_0\rangle \geq \langle v^i, s'_j\rangle\Big) \cap \bigcap_{j=1}^{p-1}\Big(\langle v^{i'}, s'_0\rangle \geq \langle v^{i'}, s'_j\rangle\Big)\right]$$

where v^i and $v^{i'}$ are vectors from the semidefinite program. Using the same argument as in the proof of Theorem 7.5, we conclude that this probability is the same for the orthogonal and simplicial porcupine models. □

We now relate the rounding procedure proposed above to the rounding procedure of Frieze and Jerrum. The first thing to notice is that the p random vectors $r_0, \dots, r_{p-1}$ are in fact close to a random orthogonal porcupine with high probability.

Lemma 7.29. *Let $\varepsilon \leq 1$. Construct the random vectors $r_0, \dots, r_{p-1}$ by choosing the components of each vector as independent $\mathrm{N}(0, 1/\sqrt{n})$ random variables. Then*

$$\mathrm{E}[\langle r_i, r_j\rangle] = \begin{cases} 1 & \text{if } i = j, \\ 0 & \text{otherwise,} \end{cases}$$
$$\Pr\big[|\langle r_i, r_j\rangle - \mathrm{E}[\langle r_i, r_j\rangle]| > \varepsilon\big] \in O\big(1/n\varepsilon^2\big).$$

Proof. If X and Y are independent $\mathrm{N}(0, 1/\sqrt{n})$ random variables,

$$\begin{aligned} \mathrm{E}[X^2] &= 1/n, \\ \mathrm{E}[X^4] &= 3/n^2, \\ \mathrm{E}[XY] &= 0, \\ \mathrm{E}[X^2Y^2] = \mathrm{E}[X^2]\,\mathrm{E}[Y^2] &= 1/n^2, \end{aligned}$$

which implies that

$$\begin{aligned} \mathrm{Var}[X^2] &= 2/n^2, \\ \mathrm{Var}[XY] &= 1/n^2. \end{aligned}$$

Since the components of the vectors $r_0, \dots, r_{p-1}$ are independent $\mathrm{N}(0, 1/\sqrt{n})$ random variables,

$$\mathrm{E}[\langle r_i, r_j \rangle] = \begin{cases} 1 & \text{if } i = j, \\ 0 & \text{otherwise,} \end{cases}$$
$$\mathrm{Var}[\langle r_i, r_j \rangle] = \begin{cases} 2/n & \text{when } i = j, \\ 1/n & \text{otherwise.} \end{cases}$$

The above equations combined with Chebyshev's inequality complete the proof. □

Note that we regard p as a constant and hide it in the $O(\cdot)$ notation. We now study the generation of the random porcupine in greater detail.

Definition 7.30. Let R be the matrix whose columns are $r_0, \dots, r_{p-1}$ and let G be the Cholesky factorization of $R^T R$, i.e., G is an upper triangular matrix such that $G^T G = R^T R$. (By construction, $R^T R$ is positive definite with probability one, and thus a unique G exists with probability one.) Define the matrix S by $S = RG^{-1}$.

Since the matrix S constructed in Definition 7.30 is an orthonormal $(n \times p)$-matrix and the matrix G used to construct S is upper triangular, multiplying R by G^{-1} from the right is equivalent to performing a Gram-Schmidt orthogonalization of the random vectors $r_0, \dots, r_{p-1}$. Thus, the vectors $s_0, \dots, s_{p-1}$, forming the columns of S, constitute a random porcupine.

Lemma 7.31. *Suppose that*

$$\big|\langle r_j, r_\ell \rangle - \mathrm{E}[\langle r_j, r_\ell \rangle]\big| \le \varepsilon \tag{7.27}$$

for all j, ℓ. Then all elements of $G - I$ are $O(\varepsilon)$.

Proof. Since the Cholesky factorization is unique for symmetric positive definite matrices, it follows from the factorization algorithm [43, Algorithm 5.2-1] that $|G_{jj} - 1| \in O(\varepsilon)$, and $|G_{j\ell}| \in O(\varepsilon)$ when $j \ne \ell$. □

Corollary 7.32. *Construct the random vectors $r_0, \dots, r_{p-1}$ by choosing the components of each vector as independent $\mathrm{N}(0, 1/\sqrt{n})$ random variables. Construct the vectors $s_0, \dots, s_{p-1}$ by performing a Gram-Schmidt orthogonalization of the vectors $r_0, \dots, r_{p-1}$. Let v be any vector in $\boldsymbol{R}^n$, and v_r be the projection of v into the subspace spanned by the vectors $r_0, \dots, r_{p-1}$. With probability at least $1 - O(1/n\varepsilon^2)$ over the choice of $r_0, \dots, r_{p-1}$,*

$$|\langle v, s_j - r_j \rangle| < \|v_r\| O(\varepsilon)$$

for all j.

Proof. Let e_j be the p-dimensional vector with zeros in all components but the jth, and let I_p be the $p \times p$ unit matrix. Then

$$\|s_j - r_j\| = \|S(I_p - G)e_j\| = \|(I_p - G)e_j\| \in O(\varepsilon),$$

with probability at least $1 - O(1/n\varepsilon^2)$ by Lemma 7.29 since, by Lemma 7.31, all elements of $I_p - G$ are $O(\varepsilon)$. □

The second important property of the rounding procedure is that the probability of a "photo finish" in the rounding procedure is small.

Lemma 7.33. *Let v be any vector in $\boldsymbol{R}^n$ and v_r be the projection of v into the subspace spanned by the vectors $r_0, \dots, r_{p-1}$. Then,*

$$\Pr[|\langle v, s_j - s_\ell \rangle| < \|v_r\|\delta] \in O(\delta).$$

Proof. By construction, the vectors $s_0, \dots, s_{p-1}$ are orthogonal unit length vectors with random orientation. Thus, we can instead view the situation as follows: We select a random unit length p-dimensional vector w from the subspace spanned by $s_0, \dots, s_{p-1}$, and compute the probability that

$$|\langle w, s \rangle| \in O(\varepsilon),$$

where $s = s_j - s_\ell$. But this probability is $O(\varepsilon)$ for any p-dimensional vector s of constant length. □

Corollary 7.34. *The probability that the edge (i, i') is not cut can be written as*

$$\sum_{j=0}^{p-1} \Pr\left[\bigcap_{\substack{\ell=0\\ \ell\neq j}}^{p-1} \left\{\langle v^i, r_j\rangle \geq \langle v^i, r_\ell\rangle\right\} \cap \bigcap_{\substack{\ell=0\\ \ell\neq j}}^{p-1} \left\{\langle v^{i'}, r_j\rangle \geq \langle v^{i'}, r_\ell\rangle\right\}\right]. \tag{7.28}$$

Suppose that

$$\left|\langle r_j, r_\ell\rangle - \mathrm{E}[\langle r_j, r_\ell\rangle]\right| \leq \varepsilon$$

for all j, ℓ. Given that (i, i') is not cut, the probability that the above inequalities hold with a margin of at least $\|v_r^i\| O(\varepsilon)$ and $\|v_r^{i'}\| O(\varepsilon)$, respectively, is $1 - O(\varepsilon)$.

Proof. By Corollary 7.32, $\langle v, r_j\rangle$ and $\langle v, s_j\rangle$ differ by at most $\|v_r\| O(\varepsilon)$ with probability $1 - O(1/n\varepsilon^2)$, and by Corollary 7.33

$$\Pr[|\langle v, s_j - s_\ell \rangle| < \|v_r\|\delta] \in O(\delta).$$

If we select $\delta \in O(\varepsilon)$ this completes the proof, since there is only a constant number of inequalities in (7.28). □

We can now fit the pieces together.

Theorem 7.35. *The probability of the edge (i, i') being cut when the solution to the semidefinite program of Frieze and Jerrum is rounded using p random vectors differs by a factor $1 + O(n^{-1/3})$ from the probability of it being cut when a random orthogonal porcupine is used in the rounding.*

Proof. It follows from Corollaries 7.32 and 7.34 that the probability that the edge (i, i') is cut when the rounding procedure uses $s_0, \ldots, s_{p-1}$ differs from the probability that it is cut when the rounding procedure uses $r_0, \ldots, r_{p-1}$ by a factor $1 - O\bigl(1/n\varepsilon^2\bigr) - O\bigl(\varepsilon\bigr)$. If we choose $\varepsilon = n^{-1/3}$ this factor is $1 - O(n^{-1/3})$. □

7.4.2 Using porcupines

Traditionally, the analysis of approximation algorithms based on semidefinite programming is done using local analysis. In our case this corresponds to finding the worst possible configuration of two porcupines (or vectors).

Theorem 7.36. *Consider the edge (i, i'). For each configuration of vectors v^i and $v^{i'}$ from Frieze and Jerrum's semidefinite program there exists a configuration of simplicial porcupines $\{u^i_j\}_{j=0}^{p-1}$ and $\{u^{i'}_j\}_{j=0}^{p-1}$ such that the ratio between the probability of the edge being cut after rounding and the corresponding term in the objective function is the same for the two configurations.*

Corollary 7.37. *Using local analysis, the performance guarantee of the porcupine algorithm for Max p-Cut is not greater than that obtained by Frieze and Jerrum.*

Proof of Theorem 7.36. We can without restriction choose coordinate system in such a way that

$$v^i = (1, 0, \ldots), \tag{7.29}$$

$$v^{i'} = (\lambda, \sqrt{1-\lambda^2}, 0, \ldots), \tag{7.30}$$

where $\lambda \geq -1/p - 1$. Let $w_j \in \boldsymbol{R}^{p-1}$, $j = 0, \ldots, p-1$, be the vertices of a regular p-simplex with $\|w_j\| = 1$. Suppose that w_j has the coordinates $w_j = (w_{j,1}, \ldots, w_{j,p-1})$, and consider a simplicial porcupine $\{u^i_j\}_{j=0}^{p-1}$ that we wish to put in correspondence with v^i. Let L_i be the $(p-1)$-dimensional subspace spanned by $\{u^i_j\}_{j=0}^{p-1}$. By symmetry, we can assume that the coordinates of u^i_j in L_i are $(w_{j,1}, \ldots, w_{j,p-1})$. We construct another simplicial porcupine $\{u^{i'}_j\}_{j=0}^{p-1}$ (corresponding to $v^{i'}$) with the following properties. Let $L_i^\perp = L_{i'} - L_i$. Denote with $\pi_L(v)$ the projection of v onto the subspace L. Then $u^{i'}_j$ can be assumed to have the coordinates $\sqrt{1-\lambda^2}(w_{j,1}, \ldots, w_{j,p-1})$ in $L_i^\perp$ (again by symmetry) and satisfy $\pi_{L_i}(u^{i'}_j) = \lambda u^i_j$. We note that $\{u^i_j\}_{j=0}^{p-1}$ and $\{u^{i'}_j\}_{j=0}^{p-1}$ satisfy the constraints (7.3a) and (7.3c).

In the rounding scheme of Frieze and Jerrum, p random vectors $r_0, \ldots, r_{p-1}$ are chosen. These vectors are n-dimensional, and all components are independent

$\mathrm{N}(0,1)$ variables. This process can be viewed as choosing a random variable from the pn-dimensional normal distribution with mean zero and unit covariance matrix.

Consider the following way to generate the random vectors $s_0, \dots, s_{p-1}$:

$$s_j = \sqrt{\frac{p-1}{p}} \sum_{\ell=1}^{p-1} w_{j,\ell} t_\ell + \sqrt{\frac{1}{p}} t_0 \tag{7.31}$$

where the components of each t_j, $j = 0, \dots, p-1$, are independent $\mathrm{N}(0,1)$. Denote with $s_{j,m}$ the mth component of s_j, for $0 \le j \le p-1$ and $1 \le m \le n$. Then $s_{j,m} \in \mathrm{N}(0,1)$ for all j and m. Furthermore, a straightforward calculation shows that

$$\mathrm{E}[s_{j,m} s_{j',m'}] = \begin{cases} 1 & \text{when } j = j' \text{ and } m = m', \\ 0 & \text{otherwise.} \end{cases}$$

Therefore the $s_{j,m}$ variables can be viewed as the components of a single random variable with the pn-dimensional normal distribution with mean zero and unit covariance matrix. This implies that rounding using the random vectors $s_0, \dots, s_{p-1}$ is equivalent to rounding using the vectors $r_0, \dots, r_{p-1}$.

Using the same techniques as in the proof of Theorem 7.5, it can be shown that we instead of the random vectors defined in (7.31) can perform the randomized rounding using the vectors

$$s'_j = \sum_{\ell=1}^{p-1} w_{j,\ell} t_\ell \tag{7.32}$$

for $j = 0, \dots, p-1$. We let $t_\ell = (\xi_\ell, \zeta_\ell, \dots)$ where $\xi_\ell, \zeta_\ell \in \mathrm{N}(0,1)$ for all ℓ. The rest of the coordinates are $\mathrm{N}(0,1)$ as well but are not used in the calculations below.

Let us now compute the probability of the edge being cut using the approach of Frieze and Jerrum. Let A^i_j be the event that $\langle v^i, s'_0 \rangle \ge \langle v^i, s'_j \rangle$. Then,

$$\begin{aligned} \Pr[(i,i') \text{ is not cut}] &= p \times \Pr[x_i \leftarrow 0 \text{ and } x_{i'} \leftarrow 0] \\ &= p \times \Pr\left[\bigcap_{j=1}^{p-1} \left(A^i_j \cap A^{i'}_j\right)\right]. \end{aligned}$$

(7.29), (7.30) and (7.32) immediately imply that

$$\begin{aligned} A^i_j &\iff \sum_{\ell=1}^{p-1} (w_{0,\ell} - w_{j,\ell}) \xi_\ell \ge 0, \\ A^{i'}_j &\iff \lambda \sum_{\ell=1}^{p-1} (w_{0,\ell} - w_{j,\ell}) \xi_\ell + \sqrt{1-\lambda^2} \sum_{\ell=1}^{p-1} (w_{0,\ell} - w_{j,\ell}) \zeta_\ell \ge 0. \end{aligned}$$

Finally, we focus on the randomized rounding used to obtain a cut from a configuration of porcupines. The random vector r used in the rounding can be assumed to satisfy

$$\pi_{L_i}(r) = (\xi_1, \xi_2, \dots, \xi_{p-1}) \tag{7.33}$$
$$\pi_{L_i^\perp}(r) = (\zeta_1, \zeta_2, \dots, \zeta_{p-1}) \tag{7.34}$$

where $\xi_i, \zeta_i \in \mathrm{N}(0,1)$ for all i. Let B_j^i be the event that $\langle u_0^i, r\rangle \geq \langle u_j^i, r\rangle$. Then,

$$\begin{aligned}\Pr[(i,i') \text{ is not cut}] &= p \times \Pr[x_i \leftarrow 0 \text{ and } x_{i'} \leftarrow 0] \\ &= p \times \Pr\left[\bigcap_{j=1}^{p-1}\left(B_j^i \cap B_j^{i'}\right)\right].\end{aligned}$$

(7.33) and (7.34) imply that

$$\begin{aligned} B_j^i &\iff \sum_{\ell=1}^{p-1}(w_{0,\ell} - w_{j,\ell})\xi_\ell \geq 0, \\ B_j^{i'} &\iff \lambda\sum_{\ell=1}^{p-1}(w_{0,\ell} - w_{j,\ell})\xi_\ell + \sqrt{1-\lambda^2}\sum_{\ell=1}^{p-1}(w_{0,\ell} - w_{j,\ell})\zeta_\ell \geq 0, \end{aligned}$$

which shows that the probability of the edge being cut is indeed the same in both cases.

To finish the proof, we just note that the corresponding terms in the objective functions in both cases evaluate to $\frac{p-1}{p}(1-\lambda)$. □

We cannot conclude that the performance guarantees are the same as there might exist porcupine configurations that cannot be put in correspondence with feasible solutions to (7.1). Also, the configurations used in the above proof might not be optimal for the semidefinite program. Using local analysis, we have obtained numerical evidence that the performance guarantees are indeed the same when $p = 3$, but we have not been able to prove it formally for any p.

Conjecture 7.38. *Using local analysis, the orthogonal and simplicial porcupine models are equivalent to Frieze and Jerrum's algorithm for Max p-Cut.*

7.5 Negative results

In this section we turn to investigating lower bounds for the Max E2-Lin mod p problem. We first construct a gadget that reduces Max E3-Lin mod p to Max E2-Lin mod p. This gadget is valid for all primes $p \geq 3$. The lower bound which follows is of the form $1 - \Theta(1/p^2)$. This lower bound can be improved by constructing a PCP that essentially reduces Max E3-Lin mod 2 to Max E2-Lin mod p. This PCP reduction is only valid for $p \geq 11$, but for those p it gives a constant lower bound, independent of p. When the two reductions are combined, it follows that there exists a universal constant such that it is **NP**-hard to approximate Max E2-Lin mod p within that factor. The details of the PCP construction are beyond the scope of this thesis, and the result is stated without proof — see [7] for details.

7.5.1 Small p

For the case $p = 2$, it is possible to use the methods of Trevisan et al. [81] to construct optimal 6-gadgets reducing PC_0 and PC_1 to Max E2-Lin mod 2. When these gadgets are combined with the hardness results by Håstad [52], it follows that it is **NP**-hard to approximate Max E2-Lin mod 2 within $11/12 + \varepsilon$. We now show how to construct a gadget which can be used to show hardness results for Max E2-Lin mod p when $p \geq 3$. Note that the gadget construction techniques of Trevisan et al. [81], described in Chapter 4, are of no use as the resulting linear programs are huge. This is the case even if one considers any reasonable restriction of the canonical witness matrix.

We start with an instance of Max E3-Lin mod p. For each equation in the instance we construct a number of equations with two variables per equation. By the result of Håstad [52], it is **NP**-hard to approximate Max E3-Lin mod p within $1/p + \varepsilon$, for any $\varepsilon > 0$, also in the special case when all coefficients in the equations are equal to one. Thus, we can assume that, for all i, the ith equation in the Max E3-Lin mod p instance is of the form

$$x_{i_1} + x_{i_2} + x_{i_3} = c.$$

For an arbitrary equation of this form we now construct the corresponding equations in the Max E2-Lin mod p instance. Consider assignments to the variables x_{i_1}, x_{i_2}, and x_{i_3} with the property that $x_{i_1} = 0$. There are p^2 such assignments, and p of those are satisfying. For each of the $p^2 - p$ unsatisfying assignments

$$(x_{i_1}, x_{i_2}, x_{i_3}) \leftarrow (0, a, b) \quad a + b \neq c$$

we introduce a new auxiliary variable $y_{i,a,b}$ and construct the following triple of equations:

$$x_{i_1} - y_{i,a,b} = 0, \tag{7.35a}$$

$$x_{i_2} - y_{i,a,b} = a, \tag{7.35b}$$

$$x_{i_3} - (p-2)y_{i,a,b} = b. \tag{7.35c}$$

There is a different $y_{i,a,b}$ for each triple. Our Max E2-Lin mod p instance contains $3m(p^2 - p)$ equations if the Max E3-Lin mod p instance contains m equations.

Lemma 7.39. *When $p \geq 3$ is prime, the above construction is a $(p-1)(p+3)$-gadget.*

Proof. Let π be an assignment to the x_i and the $y_{i,a,b}$ such that the number of satisfied equations in the Max E2-Lin mod p instance is maximized. Since each fixed $y_{i,a,b}$ occurs only in three equations, we can assume that $\pi(y_{i,a,b})$ is such that as many as possible of these three equations are satisfied. We now study some arbitrary equation

$$x_{i_1} + x_{i_2} + x_{i_3} = c \tag{7.36}$$

from the Max E3-Lin mod 2 instance, and the corresponding $3(p^2 - p)$ equations of type (7.35) from the Max E2-Lin mod p instance.

Assume that the assignment π satisfies (7.36). Then, for arbitrary a and b such that $a + b \neq c$ there is no assignment to $y_{i,a,b}$ such that all corresponding equations (7.35) containing $y_{i,a,b}$ are satisfied. For, if we sum the three equations in a triple, the left hand side becomes $x_{i_1} + x_{i_2} + x_{i_3}$ and the right hand side $a + b$. If all equations in the triple (7.35) were satisfied, then this new equation would also be satisfied. But $a + b \neq c$ by construction, which contradicts this assumption. We can, however, always satisfy one of the three equations containing $y_{i,a,b}$ by choosing $\pi(y_{i,a,b}) = \pi(x_{i_1})$. In some cases it is possible to satisfy two of the three equations. In fact, exactly $3(p-1)$ of the $p^2 - p$ triples of type (7.35) have this property. For, suppose that the satisfying assignment is

$$\pi(x_{i_1}, x_{i_2}, x_{i_3}) = (s_1, s_2, s_3).$$

Remember that each triple (7.35) corresponds to an assignment that does not satisfy (7.36). There are exactly $3(p-1)$ ways to construct unsatisfying assignments

$$\pi(x_{i_1}, x_{i_2}, x_{i_3}) = (u_{1,j}, u_{2,j}, u_{3,j})$$

with the property that (s_1, s_2, s_3) and $(u_{1,j}, u_{2,j}, u_{3,j})$ differ in exactly one position. Such an assignment corresponds to the triple

$$\begin{aligned} x_{i_1} - y_{i,a,b} &= 0, \\ x_{i_2} - y_{i,a,b} &= u_{2,j} - u_{1,j}, \\ x_{i_3} - (p-2)y_{i,a,b} &= u_{3,j} - (p-2)u_{1,j}. \end{aligned}$$

With the assignment $\pi(y_{i,a,b}) = u_{1,j}$, two of the above three equations are satisfied, since (s_1, s_2, s_3) and $(u_{1,j}, u_{2,j}, u_{3,j})$ differ in exactly one position. On the other hand, two different unsatisfying assignments $(u_{1,j}, u_{2,j}, u_{3,j})$ and $(u_{1,j'}, u_{2,j'}, u_{3,j'})$, both with the property that they differ from the satisfying assignment in exactly one position, can never correspond to the same triple. For, if that were the case, the equations

$$\begin{aligned} u_{2,j} - u_{1,j} &= u_{2,j'} - u_{1,j'} \\ u_{3,j} - (p-2)u_{1,j} &= u_{3,j'} - (p-2)u_{1,j'} \\ u_{k,j} &= u_{k,j'} \quad \text{for some } k \in \{1, 2, 3\} \end{aligned}$$

would have to be simultaneously satisfied. This, however, implies that $u_{k,j} = u_{k,j'}$ for all k. Summing up, the contribution to the objective function in the Max E2-Lin mod p instance is

$$2 \times 3(p-1) + \big((p^2 - p) - 3(p-1)\big) = (p-1)(p+3).$$

Let us now assume that the assignment π does not satisfy (7.36). Then for exactly one pair (a, b), all three equations containing $y_{i,a,b}$ can be satisfied. By a similar argument as above, exactly $3(p-2)$ of the $p^2 - p$ triples of type (7.35) have the property that two equations can be satisfied, and in the remaining triples one equation can be satisfied. The contribution to the objective function in the Max E2-Lin mod p instance is

$$\begin{aligned} &3 + 2 \times 3(p-2) + \big((p^2 - p) - (3(p-2) + 1)\big) \\ &\quad = (p-1)(p+3) - 1. \end{aligned}$$

□

Theorem 7.40. *For all $\varepsilon > 0$ and all $p \geq 3$, it is **NP**-hard to approximate Max E2-Lin mod p within $(p^2 + 3p - 1)/(p^2 + 3p) + \varepsilon$.*

Proof. By the result of Håstad [52], it is **NP**-hard to approximate Max E3-Lin mod p within $1/p + \varepsilon$, for any $\varepsilon > 0$, also in the special case when all coefficients in the equations are equal to one. When this result is combined with Lemma 7.39, the theorem follows. □

7.5.2 Large p

By modifying the PCP of Håstad [52], the following theorem is proved in [7]:

Theorem 7.41. *When $p \geq 11$, it is **NP**-hard to approximate Max E2-Lin mod p within $17/18 + \varepsilon$ for all $\varepsilon > 0$.*

When we combine this result with the results for small p, we obtain the following general result:

Theorem 7.42. *For all primes p, it is **NP**-hard to approximate Max E2-Lin mod p within $69/70 + \varepsilon$.*

Proof. For $p = 2$ we use the hardness result by Håstad [52]. For $p \in \{3, 5, 7\}$ we use Theorem 7.40, and for $p \geq 11$ we use Theorem 7.41. □

Chapter 8

An approximation algorithm for Max p-Section

8.1 Introduction

In Chapter 3, Goemans and Williamson's [40] approximation algorithm for Max Cut was described. It features a novel approach based on semidefinite programming. Frieze and Jerrum [36] extended this approach and applied it to two interesting generalizations of Max Cut: The Max p-Cut problem, where the vertices are to be partitioned into p parts instead of two, and the Max Bisection problem, where the vertices are to be partitioned into two halves of equal size. For the Max p-Cut problem they obtained a $\left(\frac{p-1}{p} + \Theta(p^{-2}\log p)\right)$-approximation algorithm and for the Max Bisection problem a 0.651-approximation algorithm. Recently, Ye [85] obtained a 0.699-approximation algorithm for the Max Bisection problem.

In this chapter we study the Max p-Section problem, which is a generalization of the Max Bisection problem: The vertices are to be partitioned into p parts of equal size so as to maximize the weight of the edges connecting different parts. Heuristics for this problem, arising in the context of finding a good layout strategy for data on parallel web servers, were recently considered by Jensch, Lüling and Sensen [53].

For Max p-Section, the approach of Frieze and Jerrum does not improve on the simple randomized algorithm: Select a p-section uniformly at random. It is easy to see that this gives a $\frac{p-1}{p}$-approximation algorithm as the probability of an edge being cut is $\frac{p-1}{p}$. Is there a way to improve on this simple algorithm? The main contribution of this chapter is a $\left(\frac{p-1}{p} + \Theta(p^{-3})\right)$-approximation algorithm for Max p-Section, thus showing that the naive randomized algorithm is not the best possible for this problem. Our algorithm is based on semidefinite programming. It is easy to formulate but the analysis is non-trivial.

Why is it harder to come up with an approximation algorithm for the Max p-Section problem than for the Max p-Cut problem? Traditionally, approximation algorithms based on semidefinite programming have been analyzed by evaluating, analytically or numerically, the performance on local configurations. For the Max p-Section problem, this technique of analysis must be amended with a global analysis to show that an even p-section is produced.

8.2 The main algorithm

The foundation of our approximation algorithm is a semidefinite relaxation of the Max p-Section problem. In Chapter 7 we considered linear equations mod p, and introduced a new type of relaxation. We will use this relaxation also in this chapter: To each vertex x_i corresponds a simplicial porcupine $\{v_j^i\}_{j=0}^{p-1}$, and to the edge (x_i, x_j) corresponds the term $\frac{p-1}{p} - \frac{p-1}{p^2}\sum_{k=0}^{p-1}\langle v_k^i, v_k^j\rangle$ (omitting the weight w_{ij}) in the objective function. If the simplices corresponding to all the vertices x_i shared vertices in $\boldsymbol{R}^n$, we could solve the Max p-Section problem to optimality using this approach. Alas, this is not the case, and we therefore add inequalities that are valid for such a configuration and which simplify the analysis. We will use the following relaxation:

$$
\begin{array}{ll}
\text{maximize} & \displaystyle\frac{p-1}{p}\sum_{i,j} w_{ij}\left(1-\frac{1}{p}\sum_{k=0}^{p-1}\langle v_k^i, v_k^j\rangle\right) \\
\text{subject to} & \langle v_k^i, v_k^i\rangle = 1 \text{ for all } i, k, \\
& \langle v_k^i, v_{k'}^i\rangle = \frac{-1}{p-1} \text{ for all } i \text{ and all } k \neq k', \\
& \langle v_k^i, v_{k'}^j\rangle \geq \frac{-1}{p-1} \text{ for all } i \neq j \text{ and all } k, k', \\
& \langle v_j^i, v_{j+k}^{i'}\rangle = \langle v_{j'}^i, v_{j'+k}^{i'}\rangle \text{ for all } i, i' \text{ and all } j, j', k, \\
& \sum_i v_k^i = 0 \text{ for all } k.
\end{array}
\tag{8.1}
$$

The first two constraints guarantee that $\{v_j^i\}_{j=0}^{p-1}$ is a simplicial porcupine, the third and fourth constraints make the solution more symmetric and therefore easier to analyze, and the last constraint encourages an even partition of the variables after the randomized rounding. For $p = 2$ the relaxation is equivalent to the relaxation used by Frieze and Jerrum [36] for the Max Bisection problem.

In the remainder of this chapter we will analyze the following randomized algorithm for the Max p-Section problem.

Algorithm 7. *Approximation algorithm for Max p-Section.*

1. *Solve the semidefinite program (8.1).*

2. *Generate* $r \in \boldsymbol{R}^{n(p-1)}$ *by choosing each component as* $N(0,1)$ *independently. For each vertex* x_i *and each* $j = 0, 1, \ldots, p-1$, *let* $q_{ij} = \frac{1}{p} + c\langle r, v_j^i\rangle$. *Now fix* i. *If all* q_{ij} *are in* $[0, 2/p]$, *set* $p_{ij} = q_{ij}$ *for all* j, *otherwise set* $p_{ij} = 1/p$ *for all* j.

3. *For each i, put x_i in part j of the partition with probability p_{ij}.*
4. *Balance the partition so that each part contains n/p vertices. This is described in detail in Section 8.4 below.*

Note that step 3 is well-defined in the sense that $\sum_j p_{ij} = 1$ for all i; this follows from the property $\sum_j v_j^i = 0$ of simplicial porcupines.

The value of the constant c depends on p; a precise expression for c will be given in Lemma 8.2 below.

An interesting feature of the algorithm is that it involves three randomized passes: First r is chosen as a random vector in $\boldsymbol{R}^{n(p-1)}$ in step 2, then a preliminary partition is constructed from independent coin tosses in step 3, and finally the balancing scheme in step 4 also makes use of randomness.

The running time of the algorithm is dominated by the time it takes to solve the semidefinite relaxation in step 1.

8.3 Analyzing local configurations

In this section we will analyze the performance of the algorithm up to step 3; the adjustments made in step 4 will be analyzed in Section 8.4.

The intuition behind the algorithm is as follows: If the porcupines $\{v_k^i\}_{k=0}^{p-1}$ and $\{v_k^j\}_{k=0}^{p-1}$ corresponding to the vertices x_i and x_j are almost perfectly misaligned, in the sense that $\langle v_k^i, v_k^j \rangle$ is close to $\frac{-1}{p-1}$, then the random variables p_{ik} and p_{jk} will be negatively correlated and the probability that x_i and x_j will be put in the same part of the partition by step 3 will be less than $1/p$. On the other hand, if the two porcupines are almost perfectly aligned, corresponding to the inner products being close to 1, the probability that the vertices will be put in the same part will be greater than $1/p$.

Consider the edge (x_i, x_j). The contribution to the objective function from this edge is

$$\frac{p-1}{p}\Big(1 - \frac{1}{p}\sum_{k=0}^{p-1} \langle v_k^i, v_k^j \rangle\Big),$$

where we omit the weight w_{ij} from now on as it does not affect the analysis. If we can bound the ratio between the probability that x_i and x_j end up in different parts and this contribution, we have a bound on the performance guarantee after step 3. We therefore set out to do just that.

Denote with $X_{ij}(r)$ the probability that the edge (x_i, x_j) is cut given the random vector r. Then the expected performance guarantee after step 3, which we will denote G, satisfies

$$G \geq \min \frac{\mathrm{E}[X_{ij}(r)]}{\frac{p-1}{p}\big(1 - \langle v_0^i, v_0^j \rangle\big)}$$

where the minimum is taken over all possible configurations of the two porcupines $\{v_k^i\}_{k=0}^{p-1}$ and $\{v_k^j\}_{k=0}^{p-1}$, and the expectation is over the choice of r. This follows from $\langle v_0^i, v_0^j\rangle = \langle v_k^i, v_k^j\rangle$ for all k, which is a consequence of the fourth constraint in the semidefinite program (8.1).

As the porcupines $\{v_k^i\}_{k=0}^{p-1}$ and $\{v_k^j\}_{k=0}^{p-1}$ together span a space of dimension at most $2(p-1)$, we will from now on assume that $r \in \boldsymbol{R}^{2(p-1)}$ for the sake of convenience. Let $\Omega_{ij} = \bigl\{r \;:\; |\langle v_k^i, r\rangle| \le 1/pc \text{ and } |\langle v_k^j, r\rangle| \le 1/pc \text{ for all } k\bigr\}$. Note that Ω_{ij} is symmetric; $r \in \Omega_{ij}$ if and only if $-r \in \Omega_{ij}$. This will simplify the analysis later on.

We can write

$$X_{ij}(r) = \begin{cases} 1 - \sum_{k=0}^{p-1} q_{ik} q_{jk} & \text{if } r \in \Omega_{ij}, \\ \frac{p-1}{p} & \text{otherwise,} \end{cases}$$

and we now turn to bounding $\mathrm{E}[X_{ij}(r)]$. As the geometry of Ω_{ij} makes it hard to calculate an exact expression, we will settle for a lower bound. This suffices as we seek to bound the performance guarantee from below.

Let $D_c = \{r \;:\; |r| \le 1/pc\}$ be the largest sphere fitting within Ω_{ij}. We now obtain

$$\begin{aligned}
\mathrm{E}[X_{ij}(r)] &= \frac{1}{(2\pi)^{p-1}} \int_{\boldsymbol{R}^{2(p-1)}} X_{ij}(r) e^{-|r|^2/2} dV \\
&\ge \frac{1}{(2\pi)^{p-1}} \int_{\Omega_{ij}} X_{ij}(r) e^{-|r|^2/2} dV \\
&\ge \frac{1}{(2\pi)^{p-1}} \int_{D_c} \Bigl(\frac{p-1}{p} - \frac{c}{p} \sum_{k=0}^{p-1} \langle v_k^i + v_k^j, r\rangle - c^2 \sum_{k=0}^{p-1} \langle v_k^i, r\rangle\langle v_k^j, r\rangle\Bigr) e^{-|r|^2/2} dV.
\end{aligned}$$

The region D_c is spherically symmetrical, and therefore $\int_{D_c} \langle v_k^i, r\rangle e^{-|r|^2/2} dV$ vanishes. This gives the bound

$$\mathrm{E}[X_{ij}(r)] \ge \frac{1}{(2\pi)^{p-1}} \int_{D_c} \Bigl(\frac{p-1}{p} - c^2 \sum_{k=0}^{p-1} \langle v_k^i, r\rangle\langle v_k^j, r\rangle\Bigr) e^{-|r|^2/2} dV. \tag{8.2}$$

We will now consider the term $\int_{D_c} \langle v_k^i, r\rangle\langle v_k^j, r\rangle e^{-|r|^2/2} dV$ in (8.2). To analyze this $2(p-1)$-dimensional integral, we introduce an orthonormal basis $\{e_i\}_{i=1}^{2(p-1)}$ for $\boldsymbol{R}^{2(p-1)}$. Without restriction, it can be chosen such that $v_k^i = e_1$ and $v_k^j = (\cos\theta)e_1 + (\sin\theta)e_2$ where θ is the angle between v_k^i and v_k^j; hence $\cos\theta = \langle v_k^i, v_k^j\rangle$. We can now write $r \in \boldsymbol{R}^{2(p-1)}$ as $r = \sum_{i=1}^{2(p-1)} r_i e_i$ with $r_i \in \boldsymbol{R}$ for all i. This gives

$$\begin{aligned}
\int_{D_c} \langle v_k^i, r\rangle\langle v_k^j, r\rangle e^{-|r|^2/2} dV &= \int_{D_c} r_1(r_1\cos\theta + r_2\sin\theta) e^{-|r|^2/2} dV \\
&= \int_{D_c} r_1^2 \langle v_k^i, v_k^j\rangle e^{-|r|^2/2} dV
\end{aligned}$$

where the last simplification is due to the symmetry of D_c.

Furthermore, as D_c is spherically symmetrical, this can be simplified further using

$$\int_{D_c} r_1^2 \langle v_k^i, v_k^j \rangle e^{-|r|^2/2} dV = \frac{1}{2(p-1)} \int_{D_c} \langle v_k^i, v_k^j \rangle |r|^2 e^{-|r|^2/2} dV.$$

Let us now turn to (8.2) again. If we combine the last sequence of simplifications with the relation $\langle v_k^i, v_k^j \rangle = \langle v_0^i, v_0^j \rangle$, valid for all k by the inequalities in the SDP (8.1), we obtain

$$\mathrm{E}[X_{ij}(r)] \geq \frac{1}{(2\pi)^{p-1}} \int_{D_c} \Big(\frac{p-1}{p} - \frac{pc^2}{2(p-1)} |r|^2 \langle v_0^i, v_0^j \rangle \Big) e^{-|r|^2/2} dV.$$

We can now exploit the symmetry of the region and the formula for the surface area of the hypersphere,

$$\int_{S^{2p-3}} dS = \frac{2\pi^{p-1}}{(p-2)!},$$

to simplify this further:

$$\begin{aligned}
\mathrm{E}[X_{ij}(r)] &\geq \frac{1}{(2\pi)^{p-1}} \int_{D_c} \Big(\frac{p-1}{p} - \frac{pc^2}{2(p-1)} |r|^2 \langle v_0^i, v_0^j \rangle \Big) e^{-|r|^2/2} dV \\
&= \frac{1}{(2\pi)^{p-1}} \int_{S^{2p-3}} dS \int_0^{1/pc} \Big(\frac{p-1}{p} - \frac{pc^2}{2(p-1)} \langle v_0^i, v_0^j \rangle s^2 \Big) s^{2p-3} e^{-s^2/2} ds \\
&= \frac{1}{2^{p-1}(p-2)!} \int_0^{1/(pc)^2} \Big(\frac{p-1}{p} - \frac{pc^2}{2(p-1)} \langle v_0^i, v_0^j \rangle u \Big) u^{p-2} e^{-u/2} du.
\end{aligned}$$

From elementary calculus we have

$$\begin{aligned}
\int u^m e^{-u/2} du &= \frac{e^{-u/2}}{-1/2} \Big(u^m - \frac{m u^{m-1}}{-1/2} + \frac{m(m-1)u^{m-2}}{(-1/2)^2} + \cdots + \frac{(-1)^m m!}{(-1/2)^m} \Big) \\
&= -2e^{-u/2} \sum_{k=0}^{m} \frac{2^{m-k} m!}{k!} u^k
\end{aligned}$$

which gives the lower bound

$$\begin{aligned}\mathrm{E}[X_{ij}(r)] \geq & \frac{1}{2^{p-1}(p-2)!}\Bigg(\frac{p-1}{p}(-2)(p-2)!\Big[e^{-u/2}\sum_{k=0}^{p-2}\frac{2^{p-2-k}u^k}{k!}\Big]_{u=0}^{u=1/(pc)^2} \\ & - \langle v_0^i, v_0^j\rangle \frac{pc^2}{2(p-1)}(-2)(p-1)!\Big[e^{-u/2}\sum_{k=0}^{p-1}\frac{2^{p-1-k}u^k}{k!}\Big]_{u=0}^{u=1/(pc)^2}\Bigg) \\ = & \frac{p-1}{p}\Big[e^{-u/2}\sum_{k=0}^{p-2}\frac{u^k}{2^k k!}\Big]_{1/(pc)^2}^{0} - pc^2\langle v_0^i, v_0^j\rangle\Big[e^{-u/2}\sum_{k=0}^{p-1}\frac{u^k}{2^k k!}\Big]_{1/(pc)^2}^{0}.\end{aligned}$$

We want to estimate the ratio between this lower bound and the contribution to the objective function; $\frac{p-1}{p}\big(1-\langle v_0^i, v_0^j\rangle\big)$. A nice feature of the lower bound on $\mathrm{E}[X_{ij}(r)]$ is that the only parameter describing the geometric relation between the two porcupines $\{v_k^i\}_{k=0}^{p-1}$ and $\{v_k^j\}_{k=0}^{p-1}$ is the inner product $\langle v_0^i, v_0^j\rangle$ which is also present in the objective function.

Lemma 8.1.

$$\frac{\frac{p-1}{p}\left[e^{-u/2}\sum_{k=0}^{p-2}\frac{u^k}{2^k k!}\right]_{1/(pc)^2}^{0} - pc^2\langle v_0^i, v_0^j\rangle\left[e^{-u/2}\sum_{k=0}^{p-1}\frac{u^k}{2^k k!}\right]_{1/(pc)^2}^{0}}{\frac{p-1}{p}\big(1-\langle v_0^i, v_0^j\rangle\big)}$$

is an increasing function in $\langle v_0^i, v_0^j\rangle$ for $\langle v_0^i, v_0^j\rangle \in \big[-\frac{1}{p-1}, 1\big)$.

Proof. The numerator is the integral of a non-negative function over a subset of $\boldsymbol{R}^{2(p-1)}$ and hence non-negative for all $\langle v_0^i, v_0^j\rangle$ in the interval and, as a special case, for $\langle v_0^i, v_0^j\rangle = 1$. This means that the fraction above can be written as

$$\frac{a - b\langle v_0^i, v_0^j\rangle}{1-\langle v_0^i, v_0^j\rangle}$$

with $a \geq b > 0$. This function is increasing for $\langle v_0^i, v_0^j\rangle \in \big[-\frac{1}{p-1}, 1\big)$. $\square$

By Lemma 8.1, we only have to consider the case $\langle v_0^i, v_0^j\rangle = -\frac{1}{p-1}$ when looking for a lower bound on the performance guarantee. This gives the lower bound

$$G \geq \frac{p-1}{p} + \frac{pc^2}{p-1} - e^{-1/2(pc)^2}\left(\frac{p-1}{p}\sum_{k=0}^{p-2}\frac{1}{(2p^2c^2)^k k!} + \frac{pc^2}{p-1}\sum_{k=0}^{p-1}\frac{1}{(2p^2c^2)^k k!}\right).$$

Out first goal is to prove that $G \geq \frac{p-1}{p} + \Theta(p^{-k})$ for some constant k. To that end we need to choose c as a function on p.

Let $c = \alpha/p^{1.5}$ where $\alpha \leq 1/2$. The last two sums in the lower bound on G are

$$\begin{aligned}
&\frac{p-1}{p}\sum_{k=0}^{p-2}\frac{1}{(2p^2c^2)^k k!} + \frac{pc^2}{p-1}\sum_{k=0}^{p-1}\frac{1}{(2p^2c^2)^k k!} \\
&= \frac{p-1}{p}\sum_{k=0}^{p-2}\frac{1}{k!}\Big(\frac{p}{2\alpha^2}\Big)^k + \frac{\alpha^2}{p^2(p-1)}\sum_{k=0}^{p-1}\frac{1}{k!}\Big(\frac{p}{2\alpha^2}\Big)^k.
\end{aligned}$$

As $2\alpha^2 \leq 1/2$, the $(k+1)$st term in each sum is at least twice as large as the kth term. Therefore

$$\begin{aligned}
&\frac{p-1}{p}\sum_{k=0}^{p-2}\frac{1}{(2p^2c^2)^k k!} + \frac{pc^2}{p-1}\sum_{k=0}^{p-1}\frac{1}{(2p^2c^2)^k k!} \\
&\leq \frac{p-1}{p}\frac{2}{(p-2)!}\Big(\frac{p}{2\alpha^2}\Big)^{p-2} + \frac{\alpha^2}{p^2(p-1)}\frac{2}{(p-1)!}\Big(\frac{p}{2\alpha^2}\Big)^{p-1} \\
&\leq \frac{2}{(p-2)!}\Big(\frac{p}{2\alpha^2}\Big)^{p-2} + \frac{1}{(p-2)!}\Big(\frac{p}{2\alpha^2}\Big)^{p-2}\cdot\frac{2\alpha^2}{p^2(p-1)^2}\Big(\frac{p}{2\alpha^2}\Big) \\
&\leq \frac{3}{(p-2)!}\Big(\frac{p}{2\alpha^2}\Big)^{p-2} \leq \frac{3p^p}{(2\alpha^2)^{p-2}p!} \leq \frac{3e^p}{\sqrt{2\pi p}(2\alpha^2)^{p-2}}
\end{aligned}$$

where the last inequality follows from the lower bound $p! \geq \sqrt{2\pi p}\,p^p e^{-p}$ from Stirling's formula. Thus

$$\begin{aligned}
G &\geq \frac{p-1}{p} + \frac{\alpha^2}{p^3} - e^{-p/2\alpha^2}\frac{3e^p}{\sqrt{2\pi p}(2\alpha^2)^{p-2}} \\
&= \frac{p-1}{p} + \frac{1}{p^3}\Big(\alpha^2 - \frac{3}{\sqrt{2\pi}}e^{p+2.5\log p-(p-2)\log(2\alpha^2)-p/2\alpha^2}\Big).
\end{aligned}$$

The following lemma is now easily verified and summarizes this section.

Lemma 8.2. *For $c = 0.2/p^{1.5}$ the performance guarantee G of Algorithm 7 when the cost of the balancing is neglected satisfies*

$$G \geq \frac{p-1}{p} + \frac{1}{30p^3}$$

for $p \geq 3$.

8.4 Balancing the partition

The partition produced in step 3 of Algorithm 7 is not necessarily an even p-section. In this section we will estimate the decrease in the objective function due to the cost of balancing and provide a balancing scheme. This concludes the analysis of the performance guarantee of Algorithm 7.

8.4.1 The distance to an even p-section

There are two types of errors that might make the partition uneven:

1. For a fixed j, $\sum_i p_{ij}$ may differ from n/p.
2. The actual number of vertices placed in part j may differ from $\sum_i p_{ij}$.

We start off by analyzing how balanced the partition can be expected to be. To that measure we let $Z_j = \sum_i p_{ij}$. For i fixed, step 2 of the algorithm sets $p_{ij} = 1/p$ for all j if any $q_{ij} = \frac{1}{p} + c\langle r, v_j^i \rangle$ is outside $[0, 2/p]$. The interval $[0, 2/p]$ is symmetric around $1/p$ and all q_{ij} have symmetric distribution around $1/p$, hence $\mathrm{E}[Z_j] = n/p$. In order to estimate the cost for balancing we need to study the variance of Z_j:

$$
\begin{aligned}
\mathrm{Var}[Z_j] &= \mathrm{E}\Big[\sum_i \sum_{i'} p_{ij} p_{i'j}\Big] - \mathrm{E}\Big[\sum_i p_{ij}\Big]^2 \\
&= \sum_i \sum_{i'} \int_{\boldsymbol{R}^{2(p-1)}} p_{ij}(r) p_{i'j}(r) \frac{e^{-|r|^2/2}}{(2\pi)^{p-1}} dV - \frac{n^2}{p^2} \\
&= \sum_i \sum_{i'} \int_{\Omega_{ii'}} p_{ij}(r) p_{i'j}(r) \frac{e^{-|r|^2/2}}{(2\pi)^{p-1}} dV \\
&\quad + \sum_i \sum_{i'} \int_{\boldsymbol{R}^{2(p-1)} \setminus \Omega_{ii'}} p_{ij}(r) p_{i'j}(r) \frac{e^{-|r|^2/2}}{(2\pi)^{p-1}} dV - \frac{n^2}{p^2}.
\end{aligned}
\tag{8.3}
$$

Consider the terms in the second sum in (8.3). We split the region into three disjoint regions as follows: Let $\boldsymbol{R}^{2(p-1)} \setminus \Omega_{ii'} = D_1 \cup D_2 \cup D_3$ where D_1 is the region where $|\langle v_k^i, r\rangle| > 1/pc$ for some k but $|\langle v_k^{i'}, r\rangle| \leq 1/pc$ for all k, D_2 is defined as D_1 but with the roles of i and i' reversed, and D_3 is the region where $|\langle v_k^i, r\rangle| > 1/pc$ for some k and $|\langle v_k^{i'}, r\rangle| > 1/pc$ for some k. By the definition of $p_{ij}(r)$, we have that $p_{ij}(r) = p_{i'j}(r) = 1/p$ for $r \in D_3$. Consider the term $p_{ij}(r)p_{i'j}(r)$ for some $r \in D_1$. Then $-r \in D_1$ as D_1 is symmetric with respect to negation. By the definition of D_1, $p_{ij}(r) = p_{ij}(-r) = 1/p$. Furthermore, $p_{i'j}(r) + p_{i'j}(-r) = (1/p + c\langle v_k^i, r\rangle) + (1/p + c\langle v_k^i, -r\rangle) = 2/p$. A similar argument can be made if $r \in D_2$. As the exponential factor only depends on $|r|$, we can conclude that

$$
\int_{\boldsymbol{R}^{2(p-1)} \setminus \Omega_{ii'}} p_{ij}(r) p_{i'j}(r) \frac{e^{-|r|^2/2}}{(2\pi)^{p-1}} dV = \int_{\boldsymbol{R}^{2(p-1)} \setminus \Omega_{ii'}} \frac{1}{p^2} \frac{e^{-|r|^2/2}}{(2\pi)^{p-1}} dV.
$$

The terms in the first sum in (8.3) are

$$
\begin{aligned}
&\int_{\Omega_{ii'}} \Big(\frac{1}{p} + c\langle r, v_j^i\rangle\Big)\Big(\frac{1}{p} + c\langle r, v_j^{i'}\rangle\Big) \frac{e^{-|r|^2/2}}{(2\pi)^{p-1}} dV \\
&= \int_{\Omega_{ii'}} \Big(\frac{1}{p^2} + c^2 \langle r, v_j^i\rangle \langle r, v_j^{i'}\rangle\Big) \frac{e^{-|r|^2/2}}{(2\pi)^{p-1}} dV
\end{aligned}
$$

where the equality follows from the symmetry of $\Omega_{ii'}$.

Combining these relations gives

$$
\begin{aligned}
&\operatorname{Var}[Z_j] \\
&= \sum_i \sum_{i'} \left(\int_{\boldsymbol{R}^{2(p-1)}} \frac{1}{p^2} \frac{e^{-|r|^2/2}}{(2\pi)^{p-1}} dV + \int_{\Omega_{ii'}} c^2 \langle r, v_j^i \rangle \langle r, v_j^{i'} \rangle \frac{e^{-|r|^2/2}}{(2\pi)^{p-1}} dV \right) - \frac{n^2}{p^2} \\
&= \sum_i \sum_{i'} \left(\mathrm{E}\Big[\frac{1}{p^2}\Big] + \int_{\Omega_{ii'}} c^2 \langle r, v_j^i \rangle \langle r, v_j^{i'} \rangle \frac{e^{-|r|^2/2}}{(2\pi)^{p-1}} dV \right) - \frac{n^2}{p^2} \\
&= \sum_i \sum_{i'} \int_{\Omega_{ii'}} c^2 \langle r, v_j^i \rangle \langle r, v_j^{i'} \rangle \frac{e^{-|r|^2/2}}{(2\pi)^{p-1}} dV.
\end{aligned}
\tag{8.4}
$$

It turns out to be beneficial to write the integral in the sum as follows:

$$
\begin{aligned}
\int_{\Omega_{ii'}} \langle r, v_j^i \rangle \langle r, v_j^{i'} \rangle \frac{e^{-|r|^2/2}}{(2\pi)^{p-1}} dV &= \int_{\boldsymbol{R}^{2(p-1)}} \langle r, v_j^i \rangle \langle r, v_j^{i'} \rangle \frac{e^{-|r|^2/2}}{(2\pi)^{p-1}} dV \\
&\quad - \int_{\boldsymbol{R}^{2(p-1)} \setminus \Omega_{ii'}} \langle r, v_j^i \rangle \langle r, v_j^{i'} \rangle \frac{e^{-|r|^2/2}}{(2\pi)^{p-1}} dV\,.
\end{aligned}
\tag{8.5}
$$

The first of these integrals has a spherically symmetric domain, so we can use the simplification from Section 8.3:

$$
\int_{\boldsymbol{R}^{2(p-1)}} \langle r, v_j^i \rangle \langle r, v_j^{i'} \rangle \frac{e^{-|r|^2/2}}{(2\pi)^{p-1}} dV = \langle v_j^i, v_j^{i'} \rangle \int_{\boldsymbol{R}^{2(p-1)}} \frac{|r|^2 e^{-|r|^2/2}}{2(p-1)(2\pi)^{p-1}} dV\,. \tag{8.6}
$$

Using the inclusion $\boldsymbol{R}^{2(p-1)} \setminus \Omega_{ii'} \subseteq \{r \,:\, |r| \geq 1/pc\}$, we can bound the size of the second integral in (8.5):

$$
\begin{aligned}
\left| \int_{\boldsymbol{R}^{2(p-1)} \setminus \Omega_{ii'}} \langle r, v_j^i \rangle \langle r, v_j^{i'} \rangle \frac{e^{-|r|^2/2}}{(2\pi)^{p-1}} dV \right| &\leq \int_{|r| \geq 1/pc} \big| \langle r, v_j^i \rangle \langle r, v_j^{i'} \rangle \big| \frac{e^{-|r|^2/2}}{(2\pi)^{p-1}} dV \\
&\leq \int_{|r| \geq 1/pc} \frac{|r|^2 e^{-|r|^2/2}}{(2\pi)^{p-1}} dV \\
&= e^{-1/2(pc)^2} 2(p-1) \sum_{k=0}^{p-1} \frac{1}{k!(2p^2c^2)^k}\,.
\end{aligned}
\tag{8.7}
$$

The last simplification follows from calculations made in Section 8.3. Notice that this rather crude estimate actually does not depend on i or i'.

We now turn to the expression (8.4) for $\mathrm{Var}[Z_j]$. Each term in the sum is decomposed into two parts in (8.5), and the sum of all terms of the first form vanishes when we use (8.6):

$$\begin{aligned}
&\sum_i \sum_{i'} \langle v_j^i, v_j^{i'} \rangle \int_{\boldsymbol{R}^{2(p-1)}} \frac{|r|^2 e^{-|r|^2/2}}{2(p-1)(2\pi)^{p-1}} dV \\
&= \sum_i \langle v_j^i, \sum_{i'} v_j^{i'} \rangle \int_{\boldsymbol{R}^{2(p-1)}} \frac{|r|^2 e^{-|r|^2/2}}{2(p-1)(2\pi)^{p-1}} dV \\
&= 0.
\end{aligned}$$

The last equality is due to the last constraint in the semidefinite program (8.1).

A bound on the sum of all terms of the second form follows immediately from (8.7), and we can conclude that

$$\mathrm{Var}[Z_j] \leq 2(p-1)c^2 n^2 e^{-1/2(pc)^2} \sum_{k=0}^{p-1} \frac{1}{k!} \Big(\frac{1}{2c^2p^2}\Big)^k.$$

As $c = \alpha/p^{1.5}$ and $\alpha \leq 1/2$, the same estimates used for the similar sums in Section 8.3 give the bound

$$\begin{aligned}
\mathrm{Var}[Z_j] &\leq 2(p-1)\frac{\alpha^2}{p^3} n^2 e^{-p/2\alpha^2} \sum_{k=0}^{p-1} \frac{1}{k!}\Big(\frac{p}{2\alpha^2}\Big)^k \\
&\leq 2(p-1)\frac{\alpha^2}{p^3} n^2 e^{-p/2\alpha^2} 2 \frac{1}{(p-1)!}\Big(\frac{p}{2\alpha^2}\Big)^{p-1} \\
&< 4n^2\alpha^2 e^{-p/2\alpha^2} \frac{p^{p-2}}{(2\alpha^2)^{p-1}p!} \\
&\leq \frac{2}{\sqrt{2\pi}} n^2 e^{-p/2\alpha^2} \frac{e^p}{p^{2.5}(2\alpha^2)^{p-2}} \\
&\leq \frac{2}{\sqrt{2\pi}} n^2 e^{p-2.5\log p - p/2\alpha^2 - (p-2)\log 2\alpha^2}.
\end{aligned}$$

It is easily verified that choosing $\alpha = 0.2$ results in $\mathrm{Var}[Z_j] \leq p^{-25}n^2$ for all $p \geq 2$.

We can now apply Chebyshev's inequality:

$$\Pr\big[|Z_j - \mathrm{E}[Z_j]| \geq an\big] \leq \frac{\mathrm{Var}[Z_j]}{(an)^2} \leq \frac{p^{-25}}{a^2}. \tag{8.8}$$

The unbalancedness due to the second kind of error is easily estimated. Let W_{ij} be the indicator variable for the event "x_i is put in part j by step 3 of the algorithm". We now apply the following Chernoff bound from [3]:

$$\Pr\left[\Big|\sum_i W_{ij} - \mathrm{E}[Z_j]\Big| > b\right] < e^{-2b^2/n} + e^{-pb^2/n}.$$

Choosing $b = n^{3/4}$, we obtain

$$\Pr\left[\left|\sum_i W_{ij} - \mathrm{E}[Z_j]\right| > n^{3/4}\right] < e^{-2n^{1/2}} + e^{-pn^{1/2}}. \tag{8.9}$$

The decrease in the objective function due to correcting this small distortion turns out to be negligible.

8.4.2 A balancing scheme and its performance

Suppose that the p parts of the partition have sizes $s_0, s_1, \dots, s_{p-1}$ after step 3 of Algorithm 7. Consider the following simple balancing scheme:

1. *$S \leftarrow \emptyset$.*
2. *For each i such that $s_i > n/p$:*
   (a) *Choose T of size $s_i - n/p$ randomly and uniformly from part i.*
   (b) *$S \leftarrow S \cup T$.*
   (c) *Remove T from part i.*
3. *For each i such that $s_i < n/p$:*
   (a) *Choose T of size $n/p - s_i$ randomly and uniformly from S.*
   (b) *$S \leftarrow S \setminus T$.*
   (c) *Add T to part i.*

How much will the objective value decrease due to balancing? Clearly the worst case is when none of the vertices being moved in step 3c of the algorithm are endpoints of any cut edges. Next we bound the cost of the balancing.

Lemma 8.3. *The expected decrease due to balancing in the expected performance guarantee for edges with at least one endpoint in part i in the partition is at most* $\max \frac{s_i - n/p}{s_i} \cdot \frac{p}{p-1}$.

Proof. Let ζ_i be the number of cut edges with one endpoint in part i of the partition. Consider the number of such edges after the set S has been formed by repeated applications of step 2a. The expected decrease compared to the number prior to the balancing algorithm being run is at most

$$\zeta_i \frac{\max\{0, s_i - n/p\}}{s_i}.$$

The lemma follows from this and the simple observation that at least a fraction $\frac{p-1}{p}$ of the edges are cut in the optimal p-section. □

The analysis from the previous section gives us the tools we need to bound the expected decrease (over the choice of r) in the expected performance guarantee due to balancing.

Theorem 8.4. *Algorithm 7 has expected performance guarantee* $\frac{p-1}{p} + \Omega(p^{-3})$ *for* $p \geq 3$ *when run with* $c = 0.2/p^{1.5}$.

Proof. If we let $a = p^{-8}$ in (8.8) and combine this equation with (8.9), we obtain

$$\Pr\bigl[\max\{s_i - n/p\} \leq p^{-8}n + n^{3/4}\bigr] \geq 1 - p \cdot \bigl(p^{-9} + e^{-2n^{1/2}} + e^{-pn^{1/2}}\bigr).$$

Lemma 8.3 can be applied when $\max\{s_i - n/p\}$ is small; otherwise the decrease in the expected performance guarantee can be bounded from above by 1. Combining Lemma 8.2 with this analysis shows that the expected performance guarantee is at least

$$\frac{p-1}{p} + \frac{1}{30p^3} - p \cdot \bigl(p^{-9} + e^{-2n^{1/2}} + e^{-pn^{1/2}}\bigr) - \frac{np^{-8} + n^{3/4}}{n/p - np^{-8} - n^{3/4}} \cdot \frac{p}{p-1}.$$

For $p \geq 3$ and large enough n, this clearly is $\frac{p-1}{p} + \Theta(p^{-3})$. □

We defer the analysis of the special case $p = 2$ (i.e., Max Bisection) to the next section.

This shows that Algorithm 7 beats the trivial randomized algorithm.

Remark 8.5. This performance guarantee, $\frac{p-1}{p} + \Theta(p^{-3})$, is somewhat weaker than the $\frac{p-1}{p} + \Theta(p^{-2}\log p)$ achieved by Frieze and Jerrum [36] for the Max p-Cut problem. It may be possible to sharpen the bound for Max p-Section but it seems hard to reach $\frac{p-1}{p} + \Theta(p^{-2})$ using the approach taken here.

8.5 Modifications for Max Bisection

One may wonder if Algorithm 7 improves on the algorithm of Frieze and Jerrum when $p = 2$. It turns out that a modified rounding scheme in step 2 improves the performance of Algorithm 7 when $p = 2$:

2'. *Generate* $r \in \boldsymbol{R}^n$ *by choosing each component as* $N(0,1)$ *independently. For each vertex* x_i *and each* $j = 0, 1, \dots, p-1$, *let* $q_{ij} = \frac{1}{p} + c\langle r, v_j^i\rangle$. *Now fix* i. *If all* q_{ij} *are in* $[0,1]$, *set* $p_{ij} = q_{ij}$ *for all* j, *otherwise set* $p_{ij} = 0$ *if* $q_{ij} \leq 0$ *and* $p_{ij} = 1$ *if* $q_{ij} \geq 1$.

As $\sum_j p_{ij} = 1$ for all i we could use only the probability that vertex x_i is put in part 0 of the partition; this is equivalent to the above construction.

For $p > 2$ this might lead to $\sum_j p_{ij} \neq 1$ for some i, but for $p = 2$ it is a generalization of Frieze and Jerrum's algorithm:

Theorem 8.6. *Frieze and Jerrum's algorithm for Max Bisection has the same worst-case behavior as the modified version of Algorithm 7 with* $c = \infty$.

Proof. The semidefinite program (8.1) is, when $p = 2$, easily seen to be equivalent to the one used by Frieze and Jerrum. Their rounding scheme corresponds to the limit $c \to \infty$ in the modified algorithm above; instead of using the inner products $\langle v_j^i, r\rangle$ to determine probabilities, they use hyperplane rounding, in which only the sign of $\langle v_j^i, r\rangle$ matters. This can be interpreted as letting $c \to \infty$. The greedy balancing scheme they use is equivalent to our probabilistic scheme in the worst case — instead of choosing T randomly as we did in the rounding scheme above, they select the T for which the decrease in the number of cut edges is minimal. When all vertices in a part result in the same decrease, this gives the same performance as selecting a random subset. □

Numerical simulations indicate that $c = \infty$ is the best choice in Algorithm 7 so our approach does not provide a better approximation algorithm for Max Bisection.

Remark 8.7. The material in this chapter first appeared in [4]. By the time this manuscript was submitted for publication, Frieze and Jerrum's algorithm was the best known for Max Bisection. Recently, Ye [85] found a better approximation algorithm, also based on semidefinite programming but using a different rounding scheme. Comparing our ideas to those of Frieze and Jerrum's has therefore become less interesting, but this section was still kept for the sake of completeness.

8.6 Lower bounds

There are no strong lower bounds on the approximability of the Max p-Section problem. The main reason for this is probably that the most successful technique for proving lower bounds, probabilistically checkable proofs (PCPs), focuses on local properties of problems. For the Max Cut problem, it has been shown that there cannot exist any approximation algorithm with performance guarantee $16/17 + \varepsilon$ for any $\varepsilon > 0$ unless $\mathbf{P} = \mathbf{NP}$ [52, 81]. PCPs have been less useful on problems with global constraints than on pure constraint satisfaction problems.

Chapter 9

Non-boolean sampling

9.1 Introduction

Arora, Karger and Karpinski [12] have constructed a randomized polynomial time approximation scheme for dense instances of a number of **Max-SNP** problems, including Max Cut. They formulate the problems as integer programs with certain properties, and then construct an algorithm finding, in probabilistic polynomial time, a solution accurate enough to give a relative error of ε, for any $\varepsilon > 0$. Fernandez de la Vega [34] has also, using other techniques, constructed a randomized polynomial time approximation scheme for dense instances of Max Cut, independently of Arora, Karger and Karpinski. It is natural to look for generalizations of these ideas to other problems. In Chapter 7 we investigated the approximability of the Max E2-Lin mod p problem, a non-binary constraint satisfaction problem. What can be said about dense instances of this and other non-binary problems? The method of Arora, Karger and Karpinski does not seem to apply in this case since the integer programs used to express such generalizations do not have the smoothness properties required by the method.

The algorithm of Fernandez de la Vega selects a random subset W of the vertices in the graph $G = (V, E)$. This subset has constant size. Then, $V \setminus W$ is partitioned randomly into smaller sets. These sets are used to construct a cut in G by exhaustive search. Finally, it turns out that the randomly selected subset W has, with high probability, the property that the cut found by the exhaustive search is close to the optimum cut in dense graphs. Goldreich, Goldwasser and Ron [42] generalize these ideas to several other problems, and express the key idea somewhat differently. In their randomized polynomial time approximation scheme for Max Cut, they partition the vertices of the graph $G = (V, E)$ into a constant number of disjoint sets V^i. For each i they find a cut in V^i by selecting a small subset U^i of the vertices in $V \setminus V^i$. Then they try all possible partitions π of U^i into two parts. Each partition π induces a cut in V^i. Finally, when π is exactly the partition from the maximum cut restricted to the subset in question, the weight of the induced cut should, with high probability, be close to the weight of the maximum cut. In this chapter, we continue this line of research by generalizing the above ideas to arbitrary non-boolean optimization problems.

Frieze and Kannan [37] have constructed a polynomial time approximation scheme for all dense **Max-SNP** problems. Their algorithm is a polynomial time approximation scheme for every problem that can be described as a general function satisfiability problem with $\Theta(n^k)$ functions. Dense instances of Max Ek-Function Sat mod p do not seem to be expressible in this manner, and on top of that, the algorithm proposed here has a simpler structure and shorter running time than their algorithm.

9.2 Systems with two variables per equation

Max E2-Lin mod p is the most natural sub-family of Max Ek-Function Sat mod p, and for clarity we will first describe the polynomial time approximation scheme and prove the results in this setting.

We consider an unweighted system of linear equations modulo some prime p. There are n variables $x_1, \dots, x_n$ in the system. The equations are of the form

$$ax_i + bx_{i'} = c$$

where $i \neq i'$, $a, b \in \boldsymbol{Z}_p^*$, and $c \in \boldsymbol{Z}_p$. We assume that there are no equivalent equations in the system. I.e., if the two equations

$$\begin{aligned} ax_i + bx_{i'} &= c \\ a'x_i + b'x_{i'} &= c' \end{aligned}$$

both are in the system, we assume that there is no $d \in \boldsymbol{Z}_p$ such that $a = da'$, $b = db'$ and $c = dc'$. We think of variable assignments as functions from the set of variables to $\boldsymbol{Z}_p$.

Definition 9.1. Denote by $S(X, \tau, x \leftarrow r)$ the number of satisfied equations with one variable from the set X and x as the other variable, given that the variables in X are assigned values according to the function τ and x is assigned the value r.

In what follows, we will sometimes regard S as a random variable over the choice of X.

The algorithm we use is based on the Max Cut algorithm by Goldreich, Goldwasser and Ron [42], and we use their terminology and notation. The parameters ℓ and t are constants independent of n. They will be determined during the analysis of the algorithm.

Algorithm 8. *A randomized polynomial time approximation scheme for dense instances of Max E2-Lin mod p.*

1. *Partition the variable set* V *into* ℓ *parts* $V^1, \dots, V^\ell$, *each of size* n/ℓ.
2. *Choose* ℓ *sets* $U^1, \dots, U^\ell$ *such that* U^i *is a set of cardinality* t *chosen uniformly at random from* $V \setminus V^i$. *Let* $U = \bigcup_{i=1}^{\ell} U^i$.
3. *For each of the (at most)* $p^{\ell t}$ *assignments* $\pi \colon U \mapsto \boldsymbol{Z}_p$, *form an assignment* $\Pi_\pi \colon V \mapsto \boldsymbol{Z}_p$ *in the following way:*
   (a) *Let* $\pi' = \pi$.
   (b) *For* $i \in \{1, \dots, \ell\}$,
   (c) *For each* $v \in V^i$,
   (d) *Let* $j^*(v)$ *be the* $j \in \boldsymbol{Z}_p$ *that maximizes* $S(U^i, \pi', v \leftarrow j)$.
   (e) *Define* $\Pi_\pi(v) = j^*(v)$.
   (f) *Modify* π' *such that* $\pi'|_{V^i} = \Pi_\pi^i$.
4. *Let* Π *be the variable assignment* Π_π *that maximizes the number of satisfied equations.*
5. *Return* Π.

The running time of the algorithm depends on how an instance is represented. A natural choice is to represent an instance as a $0/1$-table with one entry for each possible equation, where 1 corresponds to the equation being present in the instance. With this model, step 3 takes $O(n)$ time and step 4 takes $O(n^2)$ time, hence the total running time is $O(n^2)$.

Our overall goal is to show that it is possible to choose the constants ℓ and t in such a way that Algorithm 8 produces, with probability at least $1-\delta$, an assignment with weight at least $1-\varepsilon/c$ times the weight of the optimal assignment for instances with cn^2 equations. In the analysis we will use the constants ε_1 and ε_2. They are both linear in ε, and will be determined later.

The intuition behind the algorithm is as follows: Since the system of equations is dense, the sets U^i should in some sense represent the structure of $V \setminus V^i$. If we pick some variable v from V^i and some assignment to the variables in $V \setminus V^i$ we will, for each assignment $v \leftarrow j$, satisfy some fraction ϕ_j of the equations containing v and one variable from $V \setminus V^i$. We then expect U^i to have the property that the fraction of the satisfied equations containing v and one variable from U^i should be close to ϕ_j. It turns out that the decrease in the number of satisfied equations due to the sampling is $O(n^2)$, which implies that the algorithm is a polynomial time approximation scheme only for dense instances of the problem.

Let us now formalize the intuition. From now on, we fix an assignment H to the variables in V and a partition of V into ℓ parts $V^1, \dots, V^\ell$. The partition of V is arbitrary, the lexicographical order can be used, while the choice of H will be discussed later. All definitions and results proven below will be relative to these fixed entities, unless stated otherwise.

Definition 9.2. We say that the set U^i is *good* if for all, except a fraction of at most ε_1, of the variables $v \in V^i$ the inequality

$$\left| \frac{S(U^i, H, v \leftarrow j)}{t} - \frac{S(V \setminus V^i, H, v \leftarrow j)}{n - |V^i|} \right| \leq \varepsilon_2 \tag{9.1}$$

holds for all $j \in \boldsymbol{Z}_p$.

What we call a good set is essentially what Fernandez de la Vega [34] calls a *representative* set.

Lemma 9.3. *For any $\delta > 0$, it is possible to choose the constant t in such a way that the probability of a set U^i being good is at least $1 - \delta/\ell$ for a fixed i.*

Proof. Fix a variable $v \in V^i$ and some $j \in \boldsymbol{Z}_p$. Note that the assignment H is fixed; the underlying probability space is the possible choices of U^i. We now introduce, for each $w \in V \setminus V^i$, a Bernoulli random variable $\xi_{i,j,v,w}$ with the property that

$$\xi_{i,j,v,w} = \begin{cases} 1 & \text{if } w \in U^i, \\ 0 & \text{otherwise.} \end{cases}$$

We can use these random variables to express the number of satisfied equations containing v and one variable from U^i:

$$S(U^i, H, v \leftarrow j) = \sum_{w \in V \setminus V^i} S(\{w\}, H, v \leftarrow j) \xi_{i,j,v,w}. \tag{9.2}$$

Since U^i is chosen uniformly at random from $V \setminus V^i$,

$$\Pr[\xi_{i,j,v,w} = 1] = \frac{t}{n - |V^i|}. \tag{9.3}$$

Define the random variables $X_{i,j,v}$ as follows:

$$X_{i,j,v} = \frac{S(U^i, H, v \leftarrow j)}{t}.$$

From (9.2) and (9.3) it follows that

$$\mathrm{E}[X_{i,j,v}] = \sum_{w \in V \setminus V^i} \frac{S(\{w\}, H, v \leftarrow j)}{n - |V^i|} = \frac{S(V \setminus V^i, H, v \leftarrow j)}{n - |V^i|},$$

which means that we are in good shape if we can bound the probability that $X_{i,j,v}$ deviates more than ε_2 from its mean. At a first glance, this seems hard to do. For $X_{i,j,v}$ is a linear combination of dependent random variables, and the coefficients in the linear combination depend on the assignment H and the instance. Since there are, for each $w \in V \setminus V^i$, at most $p(p-1)$ equations containing v and w,

$S(\{w\}, H, v \leftarrow j)$ can be anywhere between 0 and $p-1$. Fortunately, we can use martingale tail bounds to reach our goal in spite of our limited knowledge of the probability distribution of $X_{i,j,v}$. Since the sets U^i have cardinality t, exactly t different $\xi_{i,j,v,w}$ are non-zero, which means that the sum in (9.2) can be written as

$$S(U^i, H, v \leftarrow j) = \sum_{k=1}^{t} Z_k.$$

Each random variable Z_k corresponds to $S(\{w_k\}, H, v \leftarrow j)$ for some $w_k \in V \setminus V^i$, and this implies that $0 \le Z_k \le p-1$. Thus, the sequence

$$\begin{aligned} X^0_{i,j,v} &= \mathrm{E}\left[\frac{\sum_{k=1}^t Z_k}{t}\right] \\ X^m_{i,j,v} &= \mathrm{E}\left[\frac{\sum_{k=1}^t Z_k}{t}\,\middle|\, Z_1, Z_2, \dots, Z_m\right] \quad \text{for } m = 1, \dots, t \end{aligned}$$

is a Doob martingale (see e.g. [71]) with the properties that

$$\bigl|X_{i,j,v} - \mathrm{E}[X_{i,j,v}]\bigr| = \bigl|X^t_{i,j,v} - X^0_{i,j,v}\bigr|, \tag{9.4}$$

$$\bigl|X^m_{i,j,v} - X^{m-1}_{i,j,v}\bigr| \le (p-1)/t \quad \text{for all } m \in \{1, \dots, t\}. \tag{9.5}$$

When (9.5), the so-called Lipschitz condition, is inserted into Azuma's inequality for martingales [19, 48, 71], we obtain the tail bound

$$\Pr\bigl[\bigl|X_{i,j,v} - \mathrm{E}[X_{i,j,v}]\bigr| > \varepsilon_2\bigr] < 2e^{-\varepsilon_2^2 t/2(p-1)^2}. \tag{9.6}$$

The above inequality is valid for fixed i, j and v. A set U^i is good if for all but a fraction ε_1 of the vertices in V^i, the above inequality holds for all j. Thus we keep i and j fixed and construct a new family of Bernoulli random variables

$$\eta_{i,j,v} = \begin{cases} 1 & \text{if } \bigl|X_{i,j,v} - \mathrm{E}[X_{i,j,v}]\bigr| > \varepsilon_2, \\ 0 & \text{otherwise.} \end{cases}$$

By (9.6),

$$\Pr[\eta_{i,j,v} = 1] < 2e^{-\varepsilon_2^2 t/2(p-1)^2}.$$

Furthermore,

$$\begin{aligned} \Pr[v \in V^i \text{ violates (9.1)}] &\le \sum_{j \in \boldsymbol{Z}_p} \Pr[v \in V^i \text{ violates (9.1) for } j] \\ &= \sum_{j \in \boldsymbol{Z}_p} \Pr[\eta_{ijv}] \\ &\le 2pe^{-\varepsilon_2^2 t/2(p-1)^2}. \end{aligned}$$

Let B denote the fraction of variables in V^i for which (9.1) is violated. Then

$$\mathrm{E}[B] = \frac{1}{|V^i|} \sum_{v \in V^i} \Pr[v \in V^i \text{ violates (9.1)}] \leq 2pe^{-\varepsilon_2^2 t/2(p-1)^2}$$

and applying Markov's inequality yields

$$\Pr[B \geq \varepsilon_1] \leq \frac{\mathrm{E}[B]}{\varepsilon_1} \leq \frac{2pe^{-\varepsilon_2^2 t/2(p-1)^2}}{\varepsilon_1}.$$

This implies that

$$\Pr[U^i \text{ is good}] \geq 1 - \frac{2pe^{-\varepsilon_2^2 t/2(p-1)^2}}{\varepsilon_1}.$$

Finally, we are to determine a suitable value for t, in order to make this probability large enough. If we choose

$$t \geq \frac{2(p-1)^2}{\varepsilon_2^2} \ln \frac{2\ell p}{\delta \varepsilon_1},$$

the probability that U^i is good will be at least $1 - \delta/\ell$. □

Corollary 9.4. *For any $\delta > 0$ it is possible to choose the constant t in such a way that the probability of all U^i being good is at least $1 - \delta$.*

Proof. There are ℓ different U^i. □

We construct an assignment Π to the variables in V^i by step 3 in Algorithm 8. If U^i is good, we expect the number of equations satisfied by H to be close to the number of equations satisfied by Π. To formalize this intuition, we need the following definitions.

Definition 9.5.

$$\mu(\tau) = \frac{\text{the number of equations satisfied by the assignment } \tau}{n^2}.$$

Lemma 9.6. *Let $\pi = H|_U$ and Π be the assignment produced with this choice of π in step 3 of Algorithm 8. Denote by H' the assignment that assigns to a variable v the value $H(v)$ if $v \in V \setminus V^i$ and $\Pi(v)$ if $v \in V^i$. Then, if U^i is good, it is for any $\varepsilon > 0$ possible to choose the constant ℓ in such a way that*

$$\mu(H') \geq \mu(H) - \varepsilon/p\ell.$$

Proof. We want to compare the number of equations satisfied by the assignment H to the number satisfied by H'. In particular, we want to bound the decrease in the number of satisfied equations. As only the values assigned to variables in V^i can differ between the two assignments, the possible sources of aberrations are the equations where variables in V^i are involved. We have three different cases:

1. Equations of the type $av_1 + bv_2 = c$, where $v_1, v_2 \in V^i$. There are less than $p(p-1)n^2/2\ell^2$ such equations, and at most $(p-1)n^2/2\ell^2$ of these can be satisfied by any given assignment. The decrease is thus less than $pn^2/2\ell^2$.

2. Equations of the form $av + bw = c$ where $v \in V^i$ satisfies (9.1), and $w \notin V^i$. In this case, Algorithm 8 may select the wrong assignment to v. However, that cannot decrease the number of satisfied equations by much. For, suppose that $v = j$ in the optimal solution, but the algorithm happens to set $v = j'$. The reason for that can only be that $S(U^i, \pi, v \leftarrow j') \geq S(U^i, \pi, v \leftarrow j)$. By (9.1), this implies that
$$\left| \frac{S(V \setminus V^i, H, v \leftarrow j')}{n - |V^i|} - \frac{S(V \setminus V^i, H, v \leftarrow j)}{n - |V^i|} \right| \leq 2\varepsilon_2.$$
We can therefore bound the decrease in the number of satisfied equations by
$$|V^i|\big(S(V \setminus V^i, H, v \leftarrow j') - S(V \setminus V^i, H, v \leftarrow j)\big) \leq 2\varepsilon_2 n^2/\ell.$$

3. Equations of the form $av+bw = c$ where $v \in V^i$ does not satisfy (9.1), and $w \notin V^i$. By construction there are at most $\varepsilon_1|V^i|$ such variables in V^i. The number of equations of this type is thus less than $\varepsilon_1 p^2|V^i|n$. Only $\varepsilon_1 p|V^i|n$ of these can be satisfied at the same time, and hence a bound on the decrease is $\varepsilon_1 p|V^i|n = \varepsilon_1 pn^2/\ell$.

Summing up, the total decrease is at most
$$pn^2/2\ell^2 + 2\varepsilon_2 n^2/\ell + \varepsilon_1 pn^2/\ell.$$
If we select $\ell = p^2/\varepsilon$, $\varepsilon_1 = \varepsilon/4p^2$ and $\varepsilon_2 = \varepsilon/8p$ the total decrease is at most
$$\varepsilon n^2/2p\ell + \varepsilon n^2/4p\ell + \varepsilon n^2/4p\ell = \varepsilon n^2/p\ell,$$
which concludes the proof. □

Corollary 9.7. *If all U^i are good and we construct from an assignment $\pi = H|_U$ a new assignment Π as in step 3 of Algorithm 8, it is for all $\varepsilon > 0$ possible to choose the constant ℓ in such a way that $\mu(\Pi) \geq \mu(H) - \varepsilon/p$.*

Proof. Let $H_0 = H$ and H_i, $i = 1, 2, \dots, \ell$, satisfy $H_i|_{V^i} = \Pi|_{V^i}$ and $H_i|_{V \setminus V^i} = H_{i-1}|_{V \setminus V^i}$. Apply Lemma 9.6 a total of ℓ times, once for each H_i. This corresponds to the way Algorithm 8 works. □

The only observation needed now is that since all results above are valid for *any* choice of the assignment H, they are in particular valid when H is the assignment producing the maximum number of satisfied equations.

Theorem 9.8. *For instances of Max E2-Lin mod p where the number of equations is $\Theta(n^2)$, Algorithm 8 is a randomized polynomial time approximation scheme.*

Proof. Consider the case when the assignment H is the optimal assignment. Since all possible assignments π to the variables in the set U are tried by the algorithm, the optimal assignment H restricted to U will eventually be tried. Corollaries 9.4 and 9.7 show that the algorithm produces, with probability at least $1 - \delta$, an assignment Π such that $\mu(\Pi) > \mu(H) - \varepsilon/p$. An additive error of ε/p in $\mu(\tau)$ translates into an additive error of $\varepsilon n^2/p$ for the equation problem. Since the optimum of a Max E2-Lin mod p instance with cn^2 equations is at least cn^2/p, this gives a relative error of at most ε/c. □

9.3 The general case

The algorithm described in the previous section is easily generalized to handle instances of Max Ek-Function Sat mod p as well — it does not exploit any special feature of the Max E2-Lin mod p problem. As for Max E2-Lin mod p, we assume that the set of functions does not contain any redundancy — all functions are assumed to be different. This is actually a weaker constraint than the one imposed on Max E2-Lin mod p instances; in the context of Max Ek-Function Sat mod p problems, $ax_i + bx_{i'} - c$ and $adx_i + dbx_{i'} - dc$ (for $d \notin \{0, 1\}$) are considered distinct functions whereas the corresponding Max E2-Lin mod p equations would be considered identical.

The analysis assumes that all functions in the instance are satisfiable. This is needed to ensure that the optimum of an instance with cn^k functions is at least cn^k/p^k.

We can adopt the techniques used in the proofs for the Max E2-Lin mod p case to this more general case with some minor modifications.

Definition 9.9. We extend the notation $S(X, \tau, x \leftarrow r)$ to mean the number of satisfied functions with $k - 1$ variables from the set X and one variable $x \notin X$, given that the variables in X are assigned values according to the function τ and x is assigned the value r.

Definition 9.10. We say that the set U^i is *good* if for all, except a fraction of at most ε_1, of the variables $v \in V^i$ the inequality

$$\left| \frac{S(U^i, H, v \leftarrow j)}{\binom{t}{k-1}} - \frac{S(V \setminus V^i, H, v \leftarrow j)}{\binom{n-|V^i|}{k-1}} \right| \leq \varepsilon_2 \tag{9.7}$$

holds for all $j \in \boldsymbol{Z}_p$.

Lemma 9.11. *For any $\delta > 0$, it is possible to choose the constant t in such a way that the probability of a set U^i being good is at least $1 - \delta/\ell$ for a fixed i.*

Proof. Fix a variable $v \in V^i$ and some $j \in \boldsymbol{Z}_p$. If we introduce, for each $\overline{w} \subseteq V \setminus V^i$ such that $|\overline{w}| = k-1$, a Bernoulli random variable $\xi_{i,j,v,\overline{w}}$ with the property that

$$\xi_{i,j,v,\overline{w}} = \begin{cases} 1 & \text{if } \overline{w} \subseteq U^i, \\ 0 & \text{otherwise,} \end{cases}$$

we can write

$$S(U^i, H, v \leftarrow j) = \sum_{\substack{\overline{w} \subseteq V \setminus V^i \\ |\overline{w}| = k-1}} S(\overline{w}, H, v \leftarrow j) \xi_{i,j,v,\overline{w}}. \tag{9.8}$$

There are p^{p^k} functions from $\boldsymbol{Z}_p^k$ to $\boldsymbol{Z}_p$ and a fraction $1/p$ of these are satisfied simultaneously, which implies that

$$0 \leq S(\overline{w}, H, v \leftarrow j) \leq p^{p^k - 1}.$$

To simplify the notation, we put

$$T = \binom{t}{k-1}.$$

As in the proof of Lemma 9.3, we define

$$X_{i,j,v} = \frac{S(U^i, H, v \leftarrow j)}{T}.$$

The sum in (9.8) contains T non-zero terms. We can construct a martingale $\{X_{i,j,v}^m\}_{m=0}^T$ by conditioning on these terms, as in the proof of Lemma 9.3. The Lipschitz condition in (9.5) then changes to

$$|X_{i,j,v}^m - X_{i,j,v}^{m-1}| \leq p^{p^k-1}/T \quad \text{for all } m \in \{1, \dots, T\}.$$

By choosing

$$t \geq k - 2 + \left(\frac{2(k-1)! p^{2(p^k-1)}}{\varepsilon_2^2} \ln \frac{2\ell p}{\delta \varepsilon_1} \right)^{\frac{1}{k-1}}.$$

it can be verified that the probability that U^i is good is at least $1 - \delta/\ell$. □

Lemma 9.12. *Let $\pi = H|_U$ and Π be the assignment produced with this choice of π in step 3 of Algorithm 8. Denote by H' the assignment that assigns to a variable v the value $H(v)$ if $v \in V \setminus V^i$ and $\Pi(v)$ if $v \in V^i$. Then, if U^i is good, it is for any $\varepsilon > 0$ possible to choose the constant ℓ in such a way that*

$$\mu(H') \geq \mu(H) - \varepsilon/p^k \ell.$$

Proof. To bound the decrease in the number of satisfied functions, we perform a case analysis similar to that in the proof of Lemma 9.6.

1. Functions that depend on more than one variable from V^i. At most

$$\frac{p^{p^k-1}n^k}{\ell^2}$$

such functions can be satisfied by any given assignment.

2. Functions depending on $v \in V^i$, where v satisfies (9.7), but not on any other variable in V^i. In this case Algorithm 8 can select the wrong assignment to v. Suppose that $v = j$ in the optimal solution but that the algorithm sets $v = j'$. This implies that $S(U^i, \pi, v \leftarrow j') \geq S(U^i, \pi, v \leftarrow j)$ and by (9.7),

$$\left| \frac{S(V \setminus V^i, H, v \leftarrow j')}{\binom{n-|V^i|}{k-1}} - \frac{S(V \setminus V^i, H, v \leftarrow j)}{\binom{n-|V^i|}{k-1}} \right| \leq 2\varepsilon_2.$$

The decrease in the number of satisfied functions can therefore be bounded by

$$|V^i|\big(S(V \setminus V^i, H, v \leftarrow j') - S(V \setminus V^i, H, v \leftarrow j)\big) \leq \frac{2\varepsilon_2 n^k}{(k-1)!\ell}.$$

3. Functions depending on $v \in V^i$ but not on any other variable in V^i where v does not satisfy (9.7). By construction there are at most $\varepsilon_1|V^i|$ such variables in V^i. The number of functions of this type is thus less than $\varepsilon_1|V^i|p^{p^k}n^{k-1}/(k-1)!$. Only $\varepsilon_1|V^i|p^{p^k-1}n^{k-1}/(k-1)! = \varepsilon_1 p^{p^k-1}n^k/\ell(k-1)!$ of these can be satisfied at the same time, which gives us a bound on the decrease.

Summing up, the total decrease is

$$\frac{p^{p^k-1}n^k}{\ell^2} + \frac{2\varepsilon_2 n^k}{(k-1)!\ell} + \frac{\varepsilon_1 p^{p^k-1}n^k}{(k-1)!\ell}.$$

By choosing

$$\begin{aligned} \ell &= 2p^{p^k+k-1}/\varepsilon, \\ \varepsilon_1 &= \varepsilon(k-1)!/4p^{p^k+k-1}, \\ \varepsilon_2 &= \varepsilon(k-1)!/8p^k, \end{aligned}$$

the total decrease becomes at most $\varepsilon n^k/p^k\ell$. □

Theorem 9.13. *For instances of Max Ek-Function Sat mod p where the number of satisfiable functions is $\Theta(n^k)$, Algorithm 8 is a randomized polynomial time approximation scheme.*

Proof. Follows that of Theorem 9.8 with the only difference being that the optimum of a Max Ek-Function Sat mod p instance with cn^k satisfiable functions is at least cn^k/p^k. □

9.4 Conclusions

We have shown how to construct a randomized polynomial time approximation scheme for dense instances of Max Ek-Function Sat mod p. The algorithm is intuitive, and shows the power of exhaustive sampling. The running time of the algorithm is quadratic in the number of variables, albeit with large constants. Using methods similar to those of Goldreich, Goldwasser and Ron [42], we can convert our algorithm into a randomized constant time approximation scheme. The algorithms in this scheme only compute numerical approximations to the optimum, they do not construct assignments achieving this optimum.

As a special case, Theorem 9.13 implies the existence of a polynomial time approximation scheme for dense instances of Max E3-Lin mod p. This result is interesting when compared to the lower bounds found by Håstad [52] for systems of equations with at least 3 variables in each equation: In the general case, there is no polynomial time approximation algorithm achieving a performance guarantee of $1/p+\varepsilon$ for any $\varepsilon > 0$ unless $\mathbf{P} = \mathbf{NP}$. Zwick [87] studied the problem of finding almost-satisfying assignments for almost-satisfiable instances of some constraint satisfaction problems. Also for this restricted problem, approximating Max Ek-Lin mod 2 (in the sense defined by Zwick) is hard by the results of Håstad [52].

Chapter 10

Concluding remarks

10.1 When do non-trivial approximation algorithms exist?

In this thesis we have investigated the existence of non-trivial approximation algorithms for some **NP**-hard optimization problems. Our main tool has been semidefinite relaxations, a technique which has been used extensively in a number of papers the last few years. What are the limits of this technique? Semidefinite programs are a sub-class of quadratic programs, and it is therefore no coincidence that the first applications came for Max Cut and Max 2-Sat [40]. Both these problems are constraint satisfaction problems with at most two boolean variables in each constraint. Constraints with more than two boolean variables have since been handled using canonical relaxations [86]. We have considered Max E2-Lin mod p, where each constraint contains at most two variables but the domain is $\boldsymbol{Z}_p$ instead of $\{0,1\}$. The success of semidefinite programming for this problem can be attributed to the quadratic nature of this problem: Each equation contains at most two variables. This is in contrast to Max 3-Sat and Max E3-Lin mod p for $p \geq 3$, problems for which there do not exist any non-trivial approximation algorithms unless $\mathbf{P} = \mathbf{NP}$ [52]. A common feature of these problems is that each constraint contains three variables. We tried to extend the results from Chapter 7 to arbitrary functions on $\boldsymbol{Z}_p \times \boldsymbol{Z}_p$, a class of problems for which we believe that there exist non-trivial approximation algorithms, but did not succeed.

Semidefinite programming is not limited to constraint satisfaction problems; it has also been applied to graph problems such as vertex coloring. These problems are also quadratic in nature as an edge involves two vertices. A graph problem for which semidefinite programming has not resulted in any substantial improvements is Min Vertex Cover. Finding an approximation algorithm with performance ratio a constant less than 2 is an important open problem. There are some indications that it may be hard to achieve this using traditional applications of semidefinite programming [66].

10.2 Verifiability of results based on numerical evidence

Some of the results in this thesis are based on numerical computations performed by computer programs:

- In Chapter 5, a linear program was solved in order to find the best way to combine algorithms.
- In Chapter 6, an approximation algorithm and its performance guarantee were obtained from the solution to a non-linear optimization problem. Furthermore, the gadgets and proofs of their optimality came from the solution to linear programs.
- In Chapter 7, the performance guarantee of our algorithm for the case when all equations are of the form $x_i - x_{i'} = c \bmod 3$ was calculated by solving a non-linear optimization problem.

Thus calculations made by computer programs are used to obtain theoretical results. This is a somewhat uncomfortable situation which nevertheless has been encountered numerous times in computer science and mathematics the last few decades. Several papers on approximation algorithms present results based on numerical evidence, and there are several proofs in mathematics in which computers play central roles, e.g. Appel and Haken's proof of the four-color theorem [8].

In spite of all the results above being based on calculations which would be too cumbersome to perform by hand in reasonable time, we feel that the degrees to which they can be trusted differ. This is because it for some calculations is possible to verify the numerical results by hand, while for other calculations this is out of the question.

The most easily verifiable calculation in the thesis is that in Chapter 5. There some parameters in an approximation algorithm, and the performance guarantee of the algorithm, come from the solution of a linear program. Nevertheless, it is easy to verify by hand that the so obtained algorithm achieves the claimed performance guarantee. The sharpness of the analysis is harder to verify by hand, but it would probably be straightforward to show that it cannot be improved by much.

Finding the optimal gadget reductions in Chapter 6 required considerable computing power; the largest linear program took about an hour to solve. In spite of this, it is an easy matter to verify that the gadgets achieve the claimed costs: This can be checked by trying a small number of assignments to the boolean variables involved. Proving the optimality of the gadgets is considerably harder, but still much easier than finding the gadgets. The sheer size of the problems would make it very time-consuming to do by hand, but with enough dedication it could probably be done. Something that simplifies this procedure is that the number of constraints in the gadgets, corresponding to basic variables in the optimal LP solution, is small for all reductions.

When analyzing approximation algorithms based on semidefinite programming, one way to determine the performance guarantee is to numerically estimate the minimum of a non-linear function over all configurations. How hard this is depends on how many vectors are involved in each configuration. In the first application of semidefinite programming [40], a configuration consists of two vectors, which means that it suffices to find the minimum of a function on $[0, \pi]$. Although this was done numerically, this result is easy to trust and it would be possible to prove analytically that the performance guarantee is close to that claimed. In later results, configurations have often involved more than two vectors. An example of this is our results for Max E2 Lin-3 in Chapter 7; these are based on a discretization of the parameters that are needed to describe a configuration of two porcupines containing three vectors each and one random vector. This leads to longer calculations, and consequently the results are harder to verify. When analyzing the approximation algorithm for Max 3-Horn Sat in Chapter 6, we need to analyze configurations of four vectors, for a total of six degrees of freedom. This was done by discretizing the angle space, and the calculation took several days. Notice that for all these analyses, verifying the performance guarantee is not easier than finding it in the first place — knowing which configurations achieve the minimum does not help in this respect.

The analysis of the Max 3-Horn Sat algorithm is based on numerical evidence, which does not constitute a formal mathematical proof. Karloff and Zwick encountered a similar problem with their approximation algorithm for Max 3-Sat [59]. They eventually managed to construct a long and tedious analytical proof. For Max 3-Horn Sat the situation is probably even worse as rotation functions are involved — the expressions for the separation probabilities become even more complicated. Generating a proof with the help of a computer may be possible, but such a proof might be of little value as it would probably be very hard for a human to understand. The final goal, an analytical proof that a human can understand, seems very hard to reach.

Bibliography

[1] Paola Alimonti. Non-oblivious local search for graph and hypergraph coloring problems. In *Proceedings of the 21st International Workshop on Graph-Theoretic Concepts in Computer Science*, volume 1017 of *Lecture Notes in Computer Science*, pages 167–180. Springer-Verlag, Berlin, 1995.

[2] Farid Alizadeh. Interior point methods in semidefinite programming with applications to combinatorial optimization. *SIAM Journal of Optimization*, 5:13–51, 1995.

[3] Noga Alon and Joel H. Spencer. *The Probabilistic Method.* Wiley, New York, 1992.

[4] Gunnar Andersson. An approximation algorithm for Max p-Section. In *Proceedings of the 16th Annual Symposium on Theoretical Aspects of Computer Science*, volume 1563 of *Lecture Notes in Computer Science*, pages 237–247, Berlin, 1999. Springer-Verlag, Berlin.

[5] Gunnar Andersson and Lars Engebretsen. Better approximation algorithms for Set Splitting and Not-All-Equal Sat. *Information Processing Letters*, 65:305–311, 1998.

[6] Gunnar Andersson and Lars Engebretsen. Sampling methods applied to dense instances of non-boolean optimization problems. In *2nd International Workshop on Randomization and Approximation Techniques in Computer Science*, volume 1518 of *Lecture Notes in Computer Science*, pages 357–368, Berlin, 1998. Springer-Verlag, Berlin.

[7] Gunnar Andersson, Lars Engebretsen, and Johan Håstad. A new way to use semidefinite programming with applications to linear equations mod p. In *Proceedings of the Tenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 41–50. ACM-SIAM, 1999.

[8] Kenneth Appel and Wolfgang Haken. Every planar map is four colorable. *Illinois Journal of Mathematics*, 21:429–567, 1977.

[9] Sanjeev Arora. Polynomial time approximation schemes for Euclidean TSP and other geometric problems. In *Proceedings of the 37th Annual IEEE Symposium on Foundations of Computer Science*, pages 2–11. IEEE Computer Society, Los Alamitos, 1996.

[10] Sanjeev Arora. Nearly linear time approximation schemes for Euclidean TSP and other geometric problems. In *Proceedings of the 38th Annual IEEE Symposium on Foundations of Computer Science*, pages 554–563. IEEE Computer Society, Los Alamitos, 1997.

[11] Sanjeev Arora, David R. Karger, and Marek Karpinski. Polynomial time approximation schemes for dense instances of NP-hard problems. In *Proceedings of the Twenty-seventh Annual ACM Symposium on Theory of Computing*, pages 284–293. ACM, New York, 1995.

[12] Sanjeev Arora, David R. Karger, and Marek Karpinski. Polynomial time approximation schemes for dense instances of NP-hard problems. *Journal of Computer System Sciences*, 58(1):193–210, 1999.

[13] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and hardness of approximation problems. In *Proceedings of the 33rd Annual IEEE Symposium on Foundations of Computer Science*, pages 14–23. IEEE Computer Society, Los Alamitos, 1992.

[14] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and hardness of approximation problems. *Journal of the ACM*, 45(3):501–555, 1998.

[15] Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs; a new characterization of NP. In *Proceedings of the 33rd Annual IEEE Symposium on Foundations of Computer Science*, pages 2–13. IEEE Computer Society, Los Alamitos, 1992.

[16] Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs: A new characterization of NP. *Journal of the ACM*, 45(1):70–122, 1998.

[17] Takao Asano and David P. Williamson. Improved approximation algorithms for MAX SAT. In *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms*. ACM-SIAM, 2000.

[18] Giorgio Ausiello, Pierluigi Crescenzi, Giorgio Gambosi, Viggo Kann, Alberto Marchetti-Spaccamela, and Marco Protasi. *Complexity and Approximation*. Springer-Verlag, Berlin, 1999.

[19] Kazuoki Azuma. Weighted sums of certain dependent random variables. *Tôhoku Mathematical Journal*, 19:357–367, 1967.

[20] Reuven Bar-Yehuda and Shimon Even. A linear time approximation algorithm for the weighted vertex cover problem. *Journal of Algorithms*, 2:198–203, 1981.

[21] Mihir Bellare, Oded Goldreich, and Madhu Sudan. Free bits, PCP's and non-approximability — towards tight results. *SIAM Journal of Computing*, 27(3):804–915, 1998.

[22] Stephen A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, pages 151–158. ACM, New York, 1971.

[23] Harold S.M. Coxeter. The functions of Schläfli and Lobaschefsky. *Quarterly Journal of Mathematics (Oxford)*, 6:13–29, 1935.

[24] Harold S.M. Coxeter. *Non-Euclidean geometry.* The University of Toronto Press, 1957.

[25] Pierluigi Crescenzi, Riccardo Silvestri, and Luca Trevisan. To Weight or not to Weight: Where is the Question? In *Proceedings of the 4th Israeli Symposium on Theory of Computing and Systems*, pages 68–77, 1996.

[26] Pierluigi Crescenzi and Luca Trevisan. Max NP-completeness made easy. *Theoretical Computer Science*, 225:65–79, 1999.

[27] George B. Dantzig. *Linear programming and extensions.* Princeton University Press, Princeton, 1963.

[28] George B. Dantzig, D. Ray Fulkerson, and Selmer M. Johnson. Solution of a large-scale traveling-salesman problem. *Operations Research*, 2:393–410, 1954.

[29] Jack Edmonds. Paths, trees and Flowers. *Canadian Journal of Mathematics*, 17:449–467, 1965.

[30] Ronald Fagin. Generalized first-order spectra and polynomial-time recognizable sets. In Richard M. Karp, editor, *Complexity of Computation*, volume 7, pages 43–73. SIAM AMS Proceedings, 1974.

[31] Uriel Feige and Michel X. Goemans. Approximating the value of two prover proof systems, with applications to MAX 2SAT and MAX DICUT. In *Proceedings of the 3rd Israeli Symposium on Theory of Computing and Systems*, pages 182–189, 1995.

[32] Uriel Feige, Shafi Goldwasser, László Lovász, Shmuel Safra, and Mario Szegedy. Approximating clique is almost NP-complete (preliminary version). In *Proceedings of the 32nd Annual IEEE Symposium on Foundations of Computer Science*, pages 2–12. IEEE Computer Society, Los Alamitos, 1991.

[33] Uriel Feige, Shafi Goldwasser, László Lovász, Shmuel Safra, and Mario Szegedy. Interactive proofs and the hardness of approximating cliques. *Journal of the ACM*, 43:268–292, 1996.

[34] Wenceslas Fernandez de la Vega. MAX-CUT has a randomized approximation scheme in dense graphs. *Random Structures and Algorithms*, 9:93–97, 1996.

[35] Alan M. Frieze and Mark Jerrum. Improved approximation algorithms for MAX k-CUT and MAX BISECTION. In *Proceedings of the 4th Conference on Integer Programming and Combinatorial Optimization*, volume 920 of *Lecture Notes in Computer Science*, pages 1–13, Berlin, 1995. Springer-Verlag.

[36] Alan M. Frieze and Mark Jerrum. Improved approximation algorithms for MAX k-CUT and MAX BISECTION. *Algorithmica*, 18:67–81, 1997.

[37] Alan M. Frieze and Ravi Kannan. Quick approximation to matrices and applications. *Combinatorica*, 19:175–220, 1999.

[38] Michael R. Garey and David S. Johnson. *Computers and intractability: A guide to the theory of NP-completeness.* Freeman, San Francisco, 1979.

[39] Michel X. Goemans and David P. Williamson. .878-approximation algorithms for MAX CUT and MAX 2SAT. In *Proceedings of the Twenty-sixth Annual ACM Symposium on Theory of Computing*, pages 422–431. ACM, New York, 1994.

[40] Michel X. Goemans and David P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM*, 42:1115–1145, 1995.

[41] Oded Goldreich, Shafi Goldwasser, and Dana Ron. Property testing and its connection to learning and approximation. In *Proceedings of the 37th Annual IEEE Symposium on Foundations of Computer Science*, pages 339–348. IEEE Computer Society, Los Alamitos, 1996.

[42] Oded Goldreich, Shafi Goldwasser, and Dana Ron. Property testing and its connection to learning and approximation. *Journal of the ACM*, 45(4):653–750, 1998.

[43] Gene H. Golub and Charles F. van Loan. *Matrix computations.* North Oxford Academic Publishing, Oxford, 1983.

[44] Venkatesan Guruswami. The approximability of set splitting problems and satisfiability problems with no mixed clauses. Technical Report TR99-043, Electronic Colloquium on Computational Complexity, November 1999.

[45] Eran Halperin and Uri Zwick. Approximation algorithms for MAX 4-SAT and rounding procedures for semidefinite programs. In *Proceedings of the 7th Conference on Integer Programming and Combinatorial Optimization*, volume 1610 of *Lecture Notes in Computer Science*, pages 202–217, Berlin, 1999. Springer-Verlag.

[46] Dorit S. Hochbaum. Approximation algorithms for set covering and vertex cover problems. *SIAM Journal of Computing*, 11:555–556, 1982.

[47] Dorit S. Hochbaum, editor. *Approximation algorithms for NP-hard problems.* PWS Publishing Company, Boston, 1997.

[48] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58:13–30, 1963.

[49] Wu-Yi Hsiang. On infinitesimal symmetrization and volume formula for spherical or hyperbolic tetrahedrons. *Quarterly Journal of Mathematics (Oxford)*, 39:463–468, 1988.

[50] Johan Håstad. Clique is hard to approximate within $n^{1-\varepsilon}$. In *Proceedings of the 37th Annual IEEE Symposium on Foundations of Computer Science*, pages 627–636. IEEE Computer Society, Los Alamitos, 1996.

[51] Johan Håstad. Testing of the long code and hardness for clique. In *Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing*, pages 11–19. ACM, New York, 1996.

[52] Johan Håstad. Some optimal inapproximability results. In *Proceedings of the Twenty-ninth Annual ACM Symposium on Theory of Computing*, pages 1–10. ACM, New York, 1997.

[53] Jörg Jensch, Reinhard Lüling, and Norbert Sensen. A data layout strategy for parallel web servers. In *4th International Euro-Par Conference*, volume 1470 of *Lecture Notes in Computer Science*, pages 944–952, Berlin, 1998. Springer-Verlag.

[54] David S. Johnson. Approximation algorithms for combinatorial problems. *Journal of Computer System Sciences*, 9:256–278, 1974.

[55] Viggo Kann, Jens Lagergren, and Alessandro Panconesi. Approximability of maximum splitting of k-sets and some other APX-complete problems. *Information Processing Letters*, 58:105–110, 1996.

[56] David R. Karger, Rajeev Motwani, and Madhu Sudan. Approximate graph coloring by semidefinite programming. In *Proceedings of the 35th Annual IEEE Symposium on Foundations of Computer Science*, pages 2–13, 1994.

[57] David R. Karger, Rajeev Motwani, and Madhu Sudan. Approximate graph coloring by semidefinite programming. *Journal of the ACM*, 45(2):246–265, 1998.

[58] Howard J. Karloff. How good is the Goemans-Williamson MAX CUT algorithm? In *Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing*, pages 427–434. ACM, New York, 1996.

[59] Howard J. Karloff and Uri Zwick. A 7/8-approximation algorithm for MAX 3SAT? In *Proceedings of the 38th Annual IEEE Symposium on Foundations of Computer Science*, pages 406–415. IEEE Computer Society, Los Alamitos, 1997.

[60] Narendra Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4:373–395, 1984.

[61] Narendra Karmarkar and Richard M. Karp. An efficient approximation scheme for the one-dimensional bin-packing problem. In *Proceedings of the 23rd Annual IEEE Symposium on Foundations of Computer Science*, pages 312–320. IEEE Computer Society, Los Alamitos, 1982.

[62] Richard M. Karp. Reducibility among combinatorial problems. In Raymond E. Miller and James W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, New York, NY, 1972.

[63] Ruth Kellerhals. The dilogarithm and volumes of hyperbolic polytopes. In Leonard Lewin, editor, *Structural Properties of Polylogarithms*, volume 37 of *Mathematical Surveys and Monographs*, pages 301–336. American Mathematical Society, 1991.

[64] Leonid G. Khaciyan. A polynomial algorithm for linear programming. *Doklady Akademii Nauk SSSR*, 244:1093–1096, 1979.

[65] Sanjeev Khanna, Madhu Sudan, and David P. Williamson. A complete classification of the approximability of maximization problems derived from Boolean constraint satisfaction. In *Proceedings of the Twenty-ninth Annual ACM Symposium on Theory of Computing*, pages 11–20. ACM, New York, 1997.

[66] Jon M. Kleinberg and Michel X. Goemans. The Lovász theta function and a semidefinite programming relaxation of vertex cover. *SIAM Journal of Discrete Mathematics*, 11(2), 1998.

[67] László Lovász. Coverings and colorings of hypergraphs. In *Proceedings of the 4th Southeastern Conference on Combinatorics, Graph Theory, and Computing*, pages 3–12. Utilitas Mathematica Publishing, Winnipeg, 1973.

[68] Carsten Lund, Lance Fortnow, Howard J. Karloff, and Noam Nisan. Algebraic methods for interactive proof systems. In *Proceedings of the 31st Annual IEEE Symposium on Foundations of Computer Science*, pages 2–10. IEEE Computer Society, Los Alamitos, 1990.

[69] Carsten Lund, Lance Fortnow, Howard J. Karloff, and Noam Nisan. Algebraic methods for interactive proof systems. *Journal of the ACM*, 39(4):859–868, 1992.

[70] Sanjeev Mahajan and Hariharan Ramesh. Derandomizing approximation algorithms based on semidefinite programming. *SIAM Journal of Computing*, 28(5):1641–1663, 1999.

[71] Rajeev Motwani and Prabhakar Raghavan. *Randomized algorithms.* Cambridge University Press, Cambridge, 1995.

[72] Christos H. Papadimitriou. *Computational complexity.* Addison Wesley, Reading, 1994.

[73] Christos H. Papadimitriou and Mihalis Yannakakis. Optimization, approximation, and complexity classes. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, pages 229–234. ACM, New York, 1988.

[74] Christos H. Papadimitriou and Mihalis Yannakakis. Optimization, approximation, and complexity classes. *Journal of Computer System Sciences*, 43:425–440, 1991.

[75] Erez Petrank. The hardness of approximation: Gap location. *Computational Complexity*, 4:133–157, 1994.

[76] Sartaj K. Sahni and Teofilo F. Gonzalez. P-complete approximation problems. *Journal of the ACM*, 23:555–565, 1976.

[77] Thomas J. Schaefer. The complexity of satisfiability problems. In *Proceedings of the Tenth Annual ACM Symposium on Theory of Computing*, pages 216–226. ACM, New York, 1978.

[78] Ludwig Schläfli. On the multiple integral $\int^n dx\,dy \dots dz$, whose limits are $p_1 = a_1x + b_1y + \cdots + h_1z > 0, p_2 > 0, \dots, p_n > 0$, and $x^2 + y^2 + \cdots + z^2 < 1$. *Quarterly Journal of Mathematics (Oxford)*, 2:269–300, 1858. Continued in Vol. 3 (1860), pp. 54–68 and pp. 97–108.

[79] Adi Shamir. IP = PSPACE. In *Proceedings of the 31st Annual IEEE Symposium on Foundations of Computer Science*, pages 11–15. IEEE Computer Society, Los Alamitos, 1990.

[80] Adi Shamir. IP = PSPACE. *Journal of the ACM*, 39(4):869–877, 1992.

[81] Luca Trevisan, Gregory B. Sorkin, Madhu Sudan, and David P. Williamson. Gadgets, approximation, and linear programming. In *Proceedings of the 37th Annual IEEE Symposium on Foundations of Computer Science*, pages 617–626. IEEE Computer Society, Los Alamitos, 1996.

[82] Lieven Vandenberghe and Stephen P. Boyd. Semidefinite programming. *SIAM Review*, 38:49–95, 1996.

[83] Mihalis Yannakakis. On the approximation of maximum satisfiability. In *Proceedings of the Third Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1–9. ACM-SIAM, 1992.

[84] Mihalis Yannakakis. On the approximation of maximum satisfiability. *Journal of Algorithms*, 17:475–502, 1994.

[85] Yinyu Ye. A .699-approximation algorithm for Max-Bisection. manuscript, March 1999.

[86] Uri Zwick. Approximation algorithms for constraint satisfaction problems involving at most three variables per constraint. In *Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 201–210. ACM-SIAM, 1998.

[87] Uri Zwick. Finding almost-satisfying assignments. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, pages 551–560. ACM, New York, 1998.

[88] Uri Zwick. Outward rotations: a tool for rounding solutions of semidefinite programming relaxations, with applications to MAX CUT and other problems. In *Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing*, pages 679–687. ACM, New York, 1999.