



KUNGL
TEKNISKA
HÖGSKOLAN

TRITA-NA-9809
ISSN 0348-2952
ISRN KTH/NA/R--98/9--SE

Approximating generalizations of Max Cut

Lars Engebretsen

Stockholm 1998

Licentiate's Thesis

Royal Institute of Technology

Department of Numerical Analysis and Computing Science

Akademisk avhandling som med tillstånd av Kungl Tekniska Högskolan framläggas till offentlig granskning för avläggande av teknisk licentiatexamen tisdagen den 26 maj 1998 kl 10.00 i hörsal E2, Kungl Tekniska Högskolan, Lindstedtsvägen 3, Stockholm.

© Lars Engebretsen, May 1998

Högskoletryckeriet, Stockholm 1998

Abstract

This thesis is written for the Swedish degree Licentiate of Science, *teknisk licentiat*. It is a university degree, intermediate between that of master and that of doctor.

We study the approximability of different generalizations of the Max Cut problem. First, we show that Max Set Splitting and Max Not-All-Equal Sat are both approximable within 1.380 in probabilistic polynomial time. The algorithm uses standard semidefinite relaxations, combined with a probabilistic post-processing step. Then, we focus on systems of linear equations modulo p with exactly k unknowns in each equations. The naive randomized algorithm, which guesses a solution uniformly at random from the solution space, has performance ratio p for this problem. For $k \geq 3$, it has been shown by Håstad that it is, for all $\varepsilon > 0$, **NP**-hard to approximate the optimum within $p - \varepsilon$. In contrast to this remarkable result, we show that, for the case when $k = 2$, it is possible to use a semidefinite relaxation combined with randomized rounding to obtain a probabilistic polynomial time algorithm with performance ratio $p - \kappa(p)$, where $\kappa(p) > 0$ for all p . Finally, we show that it is possible to construct a randomized polynomial time approximation scheme for instances where the number of equations is $\Theta(n^k)$, where n is the number of variables in the instance.

TRITA-NA-9809 • ISSN 0348-2952 • ISRN KTH/NA/R--98/9--SE

Contents

1	Introduction	1
1.1	A real life problem	1
1.2	Classification of problems	2
1.3	The Max Cut problem and its relatives	3
1.3.1	Set splitting	3
1.3.2	Linear equations mod p	4
1.4	Organization of the thesis	5
1.5	Acknowledgment	5
2	Set Splitting and Not-All-Equal Sat	7
2.1	Introduction	7
2.2	Basic definitions	8
2.3	The algorithms for Max Set Splitting	9
2.3.1	The algorithm of Goemans and Williamson	9
2.3.2	Our improvements	11
2.4	Analyzing the combined algorithm	12
2.4.1	Separate analyses of the contributing algorithms	12
2.4.2	The worst case for the best algorithm	14
2.5	Discussion	16
3	Linear equations mod p	17
3.1	Introduction	17
3.2	Preliminaries	18
3.2.1	Some probability theory	18
3.3	Positive results	21
3.3.1	An algorithm for equations of the form $x_i - x_{i'} = c$	21
3.3.2	General equations	26
3.4	Negative results	34
3.5	Conclusions	36

4	Denseness improves approximability	39
4.1	Preliminaries	39
4.2	Systems with two unknowns per equation	40
4.3	Systems with k unknowns per equation	46
4.4	Conclusions	51

Chapter 1

Introduction

This thesis deals with optimization problems, and their tractability. It is not difficult to realize that the size of the problem affects the tractability. The following example illustrates that it is indeed important to know how fast the time needed to solve a certain problem grows as the size of the problem increases.

1.1 A real life problem

Suppose that you are a Senior Engineer at a company producing circuit boards for computers. Naturally, you want to produce as many boards as possible each day. One phase of the production is to drill holes in the circuit board. There are n holes to be drilled and each hole takes c seconds to drill. The drill can be moved from one hole to another with velocity v . We can use these parameters to compute the total time needed to drill all holes in a board; it is

$$T = c \times n + v \times \ell, \tag{1.1}$$

where ℓ is the total length the drill must be moved from the first hole to the second, from the second to the third, and so on. To drill as many boards as possible each day, we want to minimize the time T required to drill one board. Since we must drill all n holes, it is impossible to lower the term $c \times n$, unless we acquire a new drilling machine. It is not possible to lower v since that is also a parameter that comes with the machine we use. What do we know about ℓ , the total distance the drill must move between the holes? Not much, it seems. Obviously, it depends on the order in which we drill the holes. If we have only ten holes on the board, there is a total of

$$10! = 3\,628\,800 \tag{1.2}$$

possible orders to choose. It does not take long for a computer to test all possible orders, and find the one with the shortest ℓ . But what if there are 1 000 holes on

the board? Then there are

$$1000! \approx 4 \times 10^{2567} \quad (1.3)$$

possible orders, a tremendously huge number. In this case it is infeasible to try all possible orders. Is there a better way? If we cannot come up with a better way of finding the shortest possible ℓ , is it possible to find, in reasonable time, some order, which gives a length ℓ close to the optimal one?

1.2 Classification of problems

To write down a precise definition of “solvable in reasonable time”, the notion of **NP**-completeness was introduced [7, 22]. Informally, a problem which is **NP**-complete is believed not to be solvable in reasonable time. Unfortunately, several optimization problems with wide use in applications were early shown to be **NP**-complete [14].

In applications it is often enough to know that a solution is roughly the best possible. For a problem known to be **NP**-complete, the question studied then becomes: Is it possible to find, in reasonable time, a solution close to the optimum? It turns out that the **NP**-complete problems have very different behavior with respect to approximability. The quality of an approximation algorithm can be measured with the worst possible relative error of the solution produced by the algorithm. For some problems, it is possible to find, in reasonable time, a solution with arbitrary small, but of course always positive, relative error. For some other problems, it is possible to find a solution with some constant relative error. And for some problems, it is **NP**-complete to approximate the solution within any constant relative error.

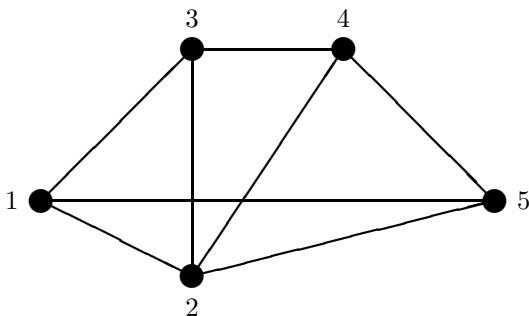
Crescenzi and Kann have constructed a list of **NP**-complete approximation problems and their approximability [8]. The list is updated continuously, and the current version (April 1997) contains more than 200 problems. For each problem, Crescenzi and Kann list *upper bounds* on the approximability, i.e., how good the best known approximation algorithm is. But they also list *lower bounds* on the approximability, i.e., bounds saying that it is impossible to approximate, in reasonable time, the solution better than some certain quantity. This reflects the research in the area: Given some optimization problem, we do not only want to know the performance of the best known approximation algorithm. We also want to know: What is the best thing we can possibly hope for? Is it worthwhile to search for better algorithms, or have we already found the best possible?

What about the problem described above? It is actually one of the most well-studied optimization problems, the Traveling Salesman Problem, or TSP for short. The traveling salesman, in our case the drill, has to visit n cities, in our case the holes in the circuit board. He wants to find, given the distances between the cities, the shortest possible tour visiting each city and starting and ending in the same

city. TSP is **NP**-complete. It is, however, possible to construct a tour of length $1 + \varepsilon$ times the length of the optimum tour in reasonable time for any $\varepsilon > 0$ [5].

1.3 The Max Cut problem and its relatives

Assume that we have a large computer network, where each computer is connected to at least one of the other computers in the network. Unfortunately, the communications protocol used in the network is very badly constructed. It requires the computers to either be in A-mode or B-mode. The problem is, that A-mode computers can only communicate with B-mode computers, and vice versa. How should we find a way to assign modes to the computers, such that as many computers as possible can communicate with each other? Let us model the network with a *graph*.



Each dot, or *vertex*, represents a computer. The lines, or *edges*, between the dots represent connections between computers. Our objective now, is to assign to each vertex the label A or B, in such a way that as many edges as possible connect an A-vertex and a B-vertex. This is exactly the Max Cut problem: Given a graph $G = (V, E)$, where V is the set of vertices and E is the set of edges, the objective in the Max Cut problem is to find the partition of V into two parts, V_1 and V_2 , which maximizes the number of edges having one endpoint in V_1 and one endpoint in V_2 . In our case, we have

$$V = \{1, 2, 3, 4, 5\}, \quad (1.4)$$

$$E = \{\{1, 2\}, \{1, 3\}, \{1, 5\}, \{2, 3\}, \{2, 4\}, \{2, 5\}, \{3, 4\}, \{4, 5\}\}. \quad (1.5)$$

In this simple case it is not difficult to see that the optimal solution is to put computers 2 and 4 in A-mode, and the other computers in B-mode. Then, six of the eight communication lines will work. In general, however, Max Cut is **NP**-complete.

1.3.1 Set splitting

By studying the Max Cut problem from two different perspectives, we obtain two natural generalizations of it. The first possibility concerns the edges in the graph.

They connect two vertices, but one could imagine edges connecting an arbitrary number of vertices. This leads to the Max Set Splitting problem: The set E above can contain not only pairs, but sets of arbitrary size. The objective is then to partition V into two parts, in such a way that as many of the sets in E as possible contain at least one member from each part in the partition. We study this problem in Chapter 2, and show that we can find, in reasonable time, a partition such that the number of sets split by this partition is at least 72% of the number of sets split in the optimal solution.

1.3.2 Linear equations mod p

We can also look at the Max Cut problem in a different way if we introduce a variable x_i for each vertex i in the graph. The variables can only assume the values 0 and 1. If we use the convention that $1 + 1 = 0$, i.e., we do our computations modulo 2, we can reformulate the Max Cut problem as a system of linear equations. The objective is then to produce an assignment to the variables x_i , such that as many equations as possible are satisfied. Our example above translates to the following system of linear equations:

$$\begin{aligned}
 x_1 + x_2 &= 1 \pmod{2} \\
 x_1 + x_3 &= 1 \pmod{2} \\
 x_1 + x_5 &= 1 \pmod{2} \\
 x_2 + x_3 &= 1 \pmod{2} \\
 x_2 + x_4 &= 1 \pmod{2} \\
 x_2 + x_5 &= 1 \pmod{2} \\
 x_3 + x_4 &= 1 \pmod{2} \\
 x_4 + x_5 &= 1 \pmod{2}
 \end{aligned} \tag{1.6}$$

A natural way to generalize this problem, is to study systems of linear equations modulo p , for arbitrary p . The system above has two variables in each equation. It has been shown by Håstad [19], that systems with exactly k variables in each equation are hard to approximate when $k \geq 3$. Specifically, it is for all $\varepsilon > 0$ impossible to satisfy more than a fraction $1/p + \varepsilon$ of the optimal number of satisfied equations, in reasonable time. On the other hand, if we guess a solution at random, we will satisfy a fraction $1/p$. We show in Chapter 3 that it is possible to satisfy more than a fraction $1/p$ of the optimum for systems with two variables in each equation. In Chapter 4, we study systems of linear equations where each variable appears many times. It turns out, that for such systems, it is possible to construct in reasonable time a solution approximating the optimum within a relative error of ε , for any $\varepsilon > 0$.

1.4 Organization of the thesis

The material in Chapter 2 follows the article by Andersson and Engebretsen [3]. My contribution to the article is approximately 50%. The material in Chapter 3 follows parts of the manuscript by Andersson, Engebretsen and Håstad [4]. My contribution to the material covered in Chapter 3 is approximately 50%. Finally, the material in Chapter 4 is based on the previously unpublished note by Andersson and Engebretsen [2]. Also in this case is my contribution approximately 50%.

1.5 Acknowledgment

The work leading up to this thesis has been carried out at the Theoretical Computer Science group at Nada, KTH. I am most grateful to my colleagues in the group, who have made it a stimulating environment to work in. Most of all, I want to thank my supervisor Viggo Kann, who has supported me in many ways during my time at the department. Also, I want to mention Gunnar Andersson, Johan Håstad and Jens Lagergren, with whom I have worked closely. Gunnar and Johan are coauthors to the papers on which the thesis is based, and Johan and Jens have never hesitated to discuss research problems with me in the most constructive way.

Chapter 2

Set Splitting and Not-All-Equal Sat

We construct new approximation algorithms for Max Set Splitting and Max Not-All-Equal Sat, which when combined with existing algorithms give the best approximation results so far for these problems. Furthermore, we solve a linear program to find an upper bound on the performance ratio. This linear program can also be used to see which of the contributing algorithms it is possible to exclude from the combined algorithm without affecting its performance ratio.

2.1 Introduction

Recently, Goemans and Williamson, building on earlier work, introduced a new paradigm for solving combinatorial optimization problems using mathematical programming [15, 17]. This significantly improved the approximability of Max Cut, and initiated a suite of yet better approximation algorithms for Max Sat and Max Cut [10, 15, 16, 17].

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 will be split, i.e., contain elements from both parts. A restriction of this problem, Max k -Set Splitting, where all subsets have cardinality k , was shown to be **NP**-complete for any fixed k by Lovász [23]. It has also been shown to be **Apx**-complete [26]. Obviously, Max 2-Set Splitting is exactly the Max Cut problem. The best known approximation algorithm for this problem has performance ratio 1.14 [15, 17]. Furthermore, Max 3-Set Splitting has been shown to be approximable within the same performance ratio [21], and for $k \geq 4$, Max k -Set Splitting is approximable within $1/(1 - 2^{1-k})$ [1, 21]. However, the

previously best known algorithm for the general Max Set Splitting problem has a performance ratio of 2.

The Max Not-All-Equal Sat problem, from now on abbreviated Max NAE Sat, is a variant 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 **Ap**-complete with performance ratio 2 [25]. 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 ideas of Crescenzi and Trevisan [9]. We start by solving a semidefinite program obtained from Goemans and Williamson [17], and then add a probabilistic postprocessing step, where the solution to the semidefinite program is perturbed. A small modification of this algorithm also gives an algorithm for Max NAE Sat. We then construct a combined algorithm by taking the maximum of the results obtained from previously known algorithms and the result from our algorithm. By formulating the inverse of the performance ratio of this combined algorithm as a linear program, we are able to bound the performance ratio simply by solving a linear program. This shows that the combined algorithm has a performance ratio of at most 1.380, both for Max Set Splitting and Max NAE Sat.

2.2 Basic definitions

Solving an **NP** maximization problem F , given an instance x , means finding a feasible solution y such that the objective value $m_F(x, y)$ is maximized. Here $m_F(x, y)$ is polynomially computable and the size of y is polynomial in $|x|$. Denote the optimal value $\max_y m_F(x, y)$ by $\text{opt}_F(x)$.

Definition 2.1. Let x be an instance of an **NP** maximization problem F and let $\text{opt}_F(x)$ be its optimum value. For any solution y to x , the performance ratio is defined as $R_F(x, y) = \text{opt}_F(x)/m_F(x, y)$.

We say that an approximation algorithm A for an **NP** maximization problem F has performance ratio $r > 1$ and performance guarantee $1/r$ if, for all input instances x , $R_F(x, A(x)) \leq r$. We say that an approximation algorithm for an **NP** maximization problem is an α -approximation algorithm if it has performance guarantee α .

Definition 2.2. 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 defined as the problem 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.

Definition 2.3. 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 defined as the problem of finding a truth assignment to the boolean variables that maximizes the total weight of the clauses C_j that contain both true and false literals.

2.3 The algorithms for Max Set Splitting

2.3.1 The algorithm of Goemans and Williamson

The basis of our work is the algorithm of Goemans and Williamson [17] for the Max Cut problem. Let $u(S_j)$ be a function which 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

$$\begin{aligned} & \text{maximize } \sum_{j=1}^n w_j z_j \\ & \text{subject to } \begin{cases} u(S_j) \geq z_j & \text{for all } j \\ 0 \leq z_j \leq 1 & \text{for all } j \end{cases} \end{aligned} \quad (2.1)$$

Now we aim to find a suitable formulation of $u(S_j)$.

Definition 2.4. For each subset S_j , we let $P_j = \{\{i_1, i_2\} : i_1 \neq i_2 \wedge i_1, i_2 \in S_j\}$.

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

$$u(S_j) = \frac{1}{|S_j| - 1} \sum_{\{i_1, i_2\} \in P_j} \frac{1 - y_{i_1} y_{i_2}}{2}. \quad (2.2)$$

We now introduce m -dimensional unit vectors v_i instead of 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 $v_{i_1} \cdot v_{i_2}$. This yields the following semidefinite program:

$$\begin{aligned} & \text{maximize } \sum_{j=1}^n w_j z_j \\ & \text{subject to } \begin{cases} \frac{1}{|S_j| - 1} \sum_{\{i_1, i_2\} \in P_j} \frac{1 - v_{i_1} \cdot v_{i_2}}{2} \geq z_j & \text{for all } j \in [1, n] \\ 0 \leq z_j \leq 1 & \text{for all } j \in [1, n] \\ v_i \cdot v_i = 1 & \text{for all } i \in [1, m] \end{cases} \end{aligned} \quad (2.3)$$

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 ρ . Then we set

$$y_i = \text{sgn}(\rho \cdot v_i). \quad (2.4)$$

To analyze the performance of this algorithm we introduce the following indicator random variables:

Definition 2.5. For each pair $\{y_{i_1}, y_{i_2}\}$ of variables, the indicator random variable $X_{i_1 i_2}$ is defined by

$$X_{i_1 i_2} = \begin{cases} 1 & \text{if } y_{i_1} \neq y_{i_2}, \\ 0 & \text{otherwise.} \end{cases} \quad (2.5)$$

Definition 2.6. For each subset S_j , the indicator random variable Z_j is defined by

$$Z_j = \begin{cases} 1 & \text{if } S_j \text{ is split by the algorithm,} \\ 0 & \text{otherwise.} \end{cases} \quad (2.6)$$

Our goal now is to show a lower bound on

$$\mathbb{E} \left[\sum_{j=1}^n w_j Z_j \right] = \sum_{j=1}^n w_j \mathbb{E}[Z_j]. \quad (2.7)$$

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 that 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

$$\Pr[X_{i_1 i_2} = 1] = \theta_{i_1 i_2} / \pi, \quad (2.8)$$

where $\theta_{i_1 i_2}$ is the angle between v_{i_1} and v_{i_2} . Goemans and Williamson show that this probability can be bounded by

$$\Pr[X_{i_1 i_2} = 1] \geq \alpha \frac{1 - v_{i_1} \cdot v_{i_2}}{2}, \quad (2.9)$$

where $\alpha > 0.87856$ [17, Sec. 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 [17, Sec. 7.2.2]:

Lemma 2.7. Define β_k as

$$\beta_k = \begin{cases} 4/k^2 & \text{if } k \text{ is even,} \\ 4/(k+1)(k-1) & \text{if } k \text{ is odd.} \end{cases} \quad (2.10)$$

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. \square

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

Theorem 2.8. For a subset S_j , we have that $E[Z_j] \geq \alpha \gamma_{|S_j|} z_j$, where

$$\gamma_{|S_j|} = \begin{cases} 4(|S_j| - 1)/|S_j|^2 & \text{if } |S_j| \text{ is even,} \\ 4/(|S_j| + 1) & \text{if } |S_j| \text{ is odd.} \end{cases} \quad (2.11)$$

Proof. By Lemma 2.7 and the linearity of expectation,

$$E[Z_j] \geq \beta_{|S_j|} \sum_{\{i_1, i_2\} \in P_j} E[X_{i_1 i_2}]. \quad (2.12)$$

Now, by the bound of Goemans and Williamson from Eq. 2.9,

$$E[X_{i_1 i_2}] \geq \alpha(1 - v_{i_1} \cdot v_{i_2})/2, \quad (2.13)$$

since $X_{i_1 i_2}$ is an indicator random variable. Thus,

$$E[Z_j] \geq \alpha \beta_{|S_j|} (|S_j| - 1) \frac{1}{|S_j| - 1} \sum_{\{i_1, i_2\} \in P_j} \frac{1 - v_{i_1} \cdot v_{i_2}}{2}. \quad (2.14)$$

By the definition of $\beta_{|S_j|}$ and $\gamma_{|S_j|}$ together with the constraints in the semidefinite program (2.3), this latter quantity is at least $\alpha \gamma_{|S_j|} z_j$. \square

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

2.3.2 Our improvements

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

Algorithm 2.9. A variant of Johnson's algorithm [20], which assigns to each $x_i \in S$ a part in the partition according to the outcome of independent tosses with a fair coin. A simple analysis shows that a set S_j is split with probability $1 - 2^{1-|S_j|}$.

Algorithm 2.10. The algorithm from Sec. 2.3.1.

Algorithm 2.10 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 [9] we construct the following algorithm:

Algorithm 2.11. For a given instance, we start by running the algorithm from Sec. 2.3.1. Then, we perturb the obtained answer by letting each x_i switch part in the partition with probability p . The value of this probability is to be specified later.

Note that if we set $p = 0$ in this algorithm we get Algorithm 2.10 above.

2.4 Analyzing the combined algorithm

As we take the maximum result obtained by any of the above algorithms, we want to bound the maximum of the contributing algorithms. Previously, such an analysis has been accomplished by constructing a new combined algorithm, which chooses as its answer the outcome of the i th 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. Since the expected value of this combined algorithm can never be greater than the maximum of the contributing algorithms, a bound on the performance of the algorithm is obtained. We will in Sec. 2.4.2 show how to use a linear program to find the q_i giving the best resulting algorithm.

2.4.1 Separate analyses of the contributing algorithms

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

We first focus on finding a lower bound on $A_{2.11}(p)$, Algorithm 2.11 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 constraint for all subset sizes that are at least N . This is reasonable as the probability that a subset S_j is split by the perturbation is high when $|S_j|$ is large.

Definition 2.12. 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 in Eq. 2.3.

Definition 2.13. Let B_j and F_j denote the events “ S_j was split before the perturbation” and “ S_j was split after the perturbation” respectively.

Lemma 2.14. $\Pr[F_j \mid B_j] \geq 1 - p(1 - p)^{|S_j|-1} - p^{|S_j|-1}(1 - p)$.

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,

$$\Pr[F_j \mid B_j] = 1 - p^\ell(1 - p)^{|S_j|-\ell} - p^{|S_j|-\ell}(1 - p)^\ell \quad (2.15)$$

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

To simplify the notation we make the following definition:

Definition 2.15. For $k < N$, let

$$\begin{aligned} r_k(p) &= \alpha\gamma_k(1 - p(1 - p)^{k-1} - p^{k-1}(1 - p)) + \\ &\quad + (1 - \alpha\gamma_k)(1 - p^k - (1 - p)^k). \end{aligned} \quad (2.16)$$

Furthermore, let

$$r_N(p) = 1 - p^N - (1 - p)^N. \quad (2.17)$$

Lemma 2.16. $\Pr[F_j] \geq z_j r_{|S_j|}(p)$ when $|S_j| < N$.

Proof. Denote by \overline{B}_j the complement of the event B_j . Using Lemma 2.14 and $\Pr[F_j \mid \overline{B}_j] = 1 - p^{|S_j|} - (1 - p)^{|S_j|}$ we obtain

$$\begin{aligned} \Pr[F_j] &= \Pr[F_j \mid B_j] \Pr[B_j] + \Pr[F_j \mid \overline{B}_j] \Pr[\overline{B}_j] \\ &\geq \Pr[B_j](1 - p(1 - p)^{|S_j|-1} - p^{|S_j|-1}(1 - p)) \\ &\quad + (1 - \Pr[B_j])(1 - p^{|S_j|} - (1 - p)^{|S_j|}) \\ &= 1 - p^{|S_j|} - (1 - p)^{|S_j|} \\ &\quad + \Pr[B_j](1 - 2p)((1 - p)^{|S_j|-1} - p^{|S_j|-1}). \end{aligned} \quad (2.18)$$

But $\Pr[B_j] \geq \alpha\gamma_{|S_j|}z_j$ (by Theorem 2.8) and $p \leq 1/2$; hence

$$\begin{aligned} \Pr[F_j] &\geq 1 - p^{|S_j|} - (1 - p)^{|S_j|} \\ &\quad + \alpha\gamma_{|S_j|}z_j(1 - 2p)((1 - p)^{|S_j|-1} - p^{|S_j|-1}) \\ &= \alpha\gamma_{|S_j|}z_j(1 - p(1 - p)^{|S_j|-1} - p^{|S_j|-1}(1 - p)) \\ &\quad + (1 - \alpha\gamma_{|S_j|}z_j)(1 - p^{|S_j|} - (1 - p)^{|S_j|}) \\ &\geq z_j r_{|S_j|}(p), \end{aligned} \quad (2.19)$$

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

Lemma 2.17. $\Pr[F_j] \geq r_N(p)$ when $|S_j| \geq N$.

Proof. We first show that $\Pr[F_j \mid B_j] \geq \Pr[F_j \mid \overline{B_j}]$.

$$\begin{aligned}
 \Pr[F_j \mid \overline{B_j}] &= 1 - (1-p)^{|S_j|} - p^{|S_j|} \\
 &= 1 - p(1-p)^{|S_j|-1} - p^{|S_j|-1}(1-p) \\
 &\quad - (1-2p)((1-p)^{|S_j|-1} - p^{|S_j|-1}) \\
 &\leq 1 - p(1-p)^{|S_j|-1} - p^{|S_j|-1}(1-p) \\
 &\leq \Pr[F_j \mid B_j].
 \end{aligned} \tag{2.20}$$

This immediately implies that

$$\begin{aligned}
 \Pr[F_j] &= \Pr[F_j \mid B_j] \Pr[B_j] + \Pr[F_j \mid \overline{B_j}] \Pr[\overline{B_j}] \\
 &\geq \Pr[F_j \mid \overline{B_j}] \\
 &= 1 - p^{|S_j|} - (1-p)^{|S_j|} \\
 &\geq r_N(p),
 \end{aligned} \tag{2.21}$$

since $|S_j| \geq N$. □

We are now able to formulate a bound on $A_{2.11}(p)$:

Theorem 2.18. $A_{2.11}(p) \geq \sum_{k=2}^{N-1} \sum_{j:|S_j|=k} w_j z_j r_k(p) + w_\infty r_N(p)$.

Proof. Follows immediately from Lemmas 2.16 and 2.17. □

The bounds for $A_{2.9}$ and $A_{2.10}$ follow from previous work [1, 17, 20, 21]:

$$A_{2.9} \geq \sum_{k=2}^{N-1} \sum_{j:|S_j|=k} w_j (1 - 2^{1-k}) + w_\infty (1 - 2^{1-N}), \tag{2.22}$$

$$A_{2.10} \geq \sum_{k=2}^{N-1} \sum_{j:|S_j|=k} w_j \alpha \gamma_k z_j. \tag{2.23}$$

2.4.2 The worst case for the best algorithm

To obtain a suitable expression for the expected value of the combination of the above algorithms, we notice that, since $0 \leq z_j \leq 1$, $A_{2.9}$ is at least

$$\sum_{k=2}^{N-1} \sum_{j:|S_j|=k} w_j z_j (1 - 2^{1-k}) + w_\infty (1 - 2^{1-N}). \tag{2.24}$$

When this is combined with Theorem 2.18 it follows that the expected weight of the solution from the combined algorithm described in Sec. 2.3.2 is at least

$$\sum_{k=2}^{N-1} \sum_{j: |S_j|=k} w_j z_j a_k + w_\infty a_N, \quad (2.25)$$

where

$$a_k = q_{-1}(1 - 2^{1-k}) + \sum_{j=0}^M q_j r_k(j/2M) \quad (2.26)$$

Here q_{-1} is the probability that Algorithm 2.9 is used while q_k , $0 \leq k \leq M$, is the probability that Algorithm 2.11 with $p = k/2M$ is used. (Notice that $p = 0$ corresponds to Algorithm 2.10.) The parameter M , which is used to discretize the probabilities p used in Algorithm 2.11, will be specified below.

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

$$\begin{aligned} & \text{maximize} \left(\min_{k \in [2, N]} a_k \right) \\ & \text{subject to} \begin{cases} \sum q_j = 1 \\ q_j \geq 0 \quad \text{for all } q_j \end{cases} \end{aligned} \quad (2.27)$$

The true contribution from the subsets S_j 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 0.724058. This means that the algorithm will always deliver a result which is at least

$$0.724058 \left(\sum_{k=2}^{N-1} \sum_{j: |S_j|=k} w_j z_j + w_\infty \right) \geq 0.724058 \cdot \text{opt}, \quad (2.28)$$

which shows that the performance ratio of our algorithm is at most 1.380. This can be compared with the best lower bound of $17/16 \approx 1.062$ [19] for the Max Cut problem. It turns out that the only non-zero q_k are q_{11} and q_{12} ; both these are approximately 0.5. Also, the term a_N in Eq. 2.26 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 . Finally, we varied M , and observed that there was no substantial improvement when we increased M above 50.

We used the publicly available package LP_SOLVE by Michel Berkelaar to solve the linear programs. 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 ratio of at most 1.380 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 with $-v_i$.

2.5 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 potential drawback is, as is indeed the case for Max Set Splitting and Max NAE Sat, that the probabilistic postprocessing destroys the good performance of the first algorithm.

The approach of expressing the performance of a combination of approximation algorithms as a linear program is quite 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, though, 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.

Chapter 3

Linear equations mod p

We introduce a new method to construct approximation algorithms for combinatorial optimization problems using semidefinite programming. The method consists of expressing each combinatorial object in the original problem as a constellation of vectors in the semidefinite program. By applying this method to systems of linear equations mod p with at most two variables in each equation, we can show that the problem is approximable within $p - \kappa(p)$, where $\kappa(p) > 0$ for all p .

3.1 Introduction

Several combinatorial maximization problems have the following property: The naive algorithm which simply chooses a solution at random from the solution space is guaranteed to give a solution of expected weight at least some constant times the weight of the optimal solution. For instance, applying the above randomized algorithm to Max Cut yields a solution with expected weight at least half the optimal weight. For a long time, better polynomial time approximation algorithms than the randomized ones were not known to exist for many of the problems with the above property. This situation changed when Goemans and Williamson [17] showed that it is possible to use semidefinite programming to approximate Max Cut and Max 2-Sat within 1.14. Extending the techniques of Goemans and Williamson, Frieze and Jerrum [12] showed that it is possible to construct a polynomial time approximation algorithm, which is better than the simple randomized one, also for Max k -Cut.

Systems of linear equations mod p is a basic and very general combinatorial problem, which exhibits the property described above: The naive randomized algorithm which chooses a solution at random approximates the problem within p . Recently, Håstad [19] studied systems of linear equations mod p with exactly k unknowns in each equation, and showed that it is actually **NP**-hard to approximate the problem within $p - \varepsilon$ for all $\varepsilon > 0$, all primes p , and all $k \geq 3$.

We study another problem of this type, systems of linear equations mod p with at most two unknowns in each equation. When $p = 2$, this problem has been extensively studied, but for $p > 2$ not much is known. We use semidefinite programming combined with randomized rounding to show, that for systems of linear equations mod p with at most two variables per equation it is possible to do better than the naive randomized heuristic. Specifically, we show that there exists, for all primes p , a randomized polynomial time algorithm approximating the problem within $p - \kappa(p)$, where $\kappa(p) > 0$ for all p .

The usual way to use semidefinite programming in approximation algorithms was, at first, to formulate the problem as an integer program and then relax this program to a semidefinite one. In order to approximate Max k -Cut, Frieze and Jerrum [12] 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 type of symmetries as the solution to the original problem.

3.2 Preliminaries

Definition 3.1. We denote by Max Ek -Lin mod p the problem of, given a system of linear equations mod p with exactly k variables in each equation, maximizing the number of satisfied equations.

Definition 3.2. We denote by Max k -Lin mod p the problem of, given a system of linear equations mod p with at most k variables in each equation, maximizing the number of satisfied equations.

From now on, p always denotes a prime, although all our results generalize to composite p .

3.2.1 Some probability theory

We will now formulate and prove some lemmas, which are needed later in the paper. The proofs use only basic probability theory; they are included here for the sake of completeness. Let

$$\varphi(x) = \frac{e^{-x^2/2}}{\sqrt{2\pi}} \tag{3.1}$$

and

$$\Phi(x) = \int_{-\infty}^x \varphi(t) dt. \tag{3.2}$$

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

$$\begin{aligned}
 \sqrt{2\pi}(1 - \Phi(x)) &= \int_x^\infty e^{-t^2/2} dt \\
 &= \int_x^\infty t e^{-t^2/2} t^{-1} dt \\
 &= \frac{e^{-x^2/2}}{x} - \int_x^\infty e^{-t^2/2} t^{-2} dt \\
 &= \left(\frac{1}{x} - \frac{1}{x^3} \right) e^{-x^2/2} + 3 \int_x^\infty e^{-t^2/2} t^{-4} dt.
 \end{aligned} \tag{3.3}$$

The above equalities immediately imply, that

$$\varphi(x) \left(\frac{1}{x} - \frac{1}{x^3} \right) < 1 - \Phi(x) < \frac{\varphi(x)}{x}, \tag{3.4}$$

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

Lemma 3.3. *Let X_0, \dots, X_{p-1} be i.i.d. $N(0, 1)$ random variables. Denote the maximum of the X_i by $X_{(p)}$, and the second maximum by $X_{(p-1)}$. Then, for any $\delta > 0$,*

$$\begin{aligned}
 \Pr \left[X_{(p)} \geq (1 + \delta) \sqrt{2 \ln p} \cap X_{(p-1)} \leq (1 + \delta/2) \sqrt{2 \ln p} \right] \\
 > \frac{1}{2p^{2\delta+\delta^2}(1+\delta)\sqrt{\pi \ln p}} \left(1 - \frac{1}{2 \ln p} - \frac{1}{2p^\delta \sqrt{\pi \ln p}} \right).
 \end{aligned} \tag{3.5}$$

Proof. Since the X_i are i.i.d. $N(0, 1)$,

$$\Pr[X_{(p)} \geq x \cap X_{(p-1)} \leq y] = p(1 - \Phi(x))\Phi(y)^{p-1}. \tag{3.6}$$

when $x \geq y$. We now apply the bound on $\Phi(x)$ from Eq. 3.4. This bound, together with the fact that $\delta > 0$, implies that

$$\begin{aligned}
 1 - \Phi \left((1 + \delta) \sqrt{2 \ln p} \right) &> \frac{1}{\sqrt{2\pi}p^{(1+\delta)^2}} \left(\frac{1}{(1 + \delta) \sqrt{2 \ln p}} \right. \\
 &\quad \left. - \frac{1}{(1 + \delta)^3 (2 \ln p)^{3/2}} \right) \\
 &> \frac{1}{2p^{1+2\delta+\delta^2}(1+\delta)\sqrt{\pi \ln p}} \left(1 - \frac{1}{2 \ln p} \right),
 \end{aligned} \tag{3.7}$$

and that

$$\begin{aligned}
 \Phi \left((1 + \delta/2) \sqrt{2 \ln p} \right) &> 1 - \frac{1}{\sqrt{2\pi}p^{(1+\delta/2)^2}(1+\delta/2)\sqrt{2 \ln p}} \\
 &> 1 - \frac{1}{2p^{1+\delta}\sqrt{\pi \ln p}}.
 \end{aligned} \tag{3.8}$$

When this is inserted in Eq. 3.6, the lemma follows. \square

Lemma 3.4. *Let X and Z be i.i.d. $N(0, 1)$ and $\varepsilon \in]0, 1[$. Then, for any $\delta > 0$,*

$$\begin{aligned} \Pr \left[\left| \left(1 - \sqrt{1 - \varepsilon}\right)X - \sqrt{\varepsilon}Z \right| > \frac{\delta}{4} \sqrt{2(1 - \varepsilon) \ln p} \right] \\ \leq \frac{4p^{-\delta^2(1-\varepsilon)/32\varepsilon}}{\delta} \sqrt{\frac{2\varepsilon}{(1 - \varepsilon)\pi \ln p}}. \end{aligned} \quad (3.9)$$

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

$$\sigma = \sqrt{\left(1 - \sqrt{1 - \varepsilon}\right)^2 + \varepsilon} \leq \sqrt{2\varepsilon}. \quad (3.10)$$

Since $\Pr[|Y| > y] = 2(1 - \Phi(y/\sigma))$, we can use Eq. 3.4.

$$\begin{aligned} \Pr \left[|Y| > \frac{\delta}{4} \sqrt{2(1 - \varepsilon) \ln p} \right] &= 2 \left(1 - \Phi \left(\frac{\delta}{4\sigma} \sqrt{2(1 - \varepsilon) \ln p} \right) \right) \\ &\leq 2 \times \frac{4\sigma}{\delta \sqrt{2(1 - \varepsilon) \ln p}} \times \frac{p^{-\delta^2(1-\varepsilon)/16\sigma^2}}{\sqrt{2\pi}} \\ &\leq \frac{4p^{-\delta^2(1-\varepsilon)/32\varepsilon}}{\delta} \sqrt{\frac{2\varepsilon}{(1 - \varepsilon)\pi \ln p}}. \end{aligned} \quad (3.11)$$

□

Lemma 3.5. *Let X_0, \dots, X_{p-1} be i.i.d. $N(0, 1)$ random variables. Denote the maximum of the X_i by $X_{(p)}$, and the second maximum by $X_{(p-1)}$. Then*

$$\Pr \left[X_{(p)} > X_{(p-1)} + \delta \right] > 1 - \frac{p^2 \delta}{(p-1) \sqrt{2\pi}}. \quad (3.12)$$

Proof. Since the X_i are independent,

$$\Pr \left[X_{(p)} > X_{(p-1)} + \delta \right] = p \times \Pr \left[\bigcap_{i=1}^{p-1} X_0 > X_i + \delta \right]. \quad (3.13)$$

To compute the latter probability we condition on X_0 .

$$\Pr \left[\bigcap_{i=1}^{p-1} X_0 > X_i + \delta \right] = \int_{-\infty}^{\infty} \Phi^{p-1}(x - \delta) \varphi(x) dx. \quad (3.14)$$

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

$$\begin{aligned} \Phi^{p-1}(x - \delta) &= (\Phi(x) - \delta \varphi(\xi))^{p-1} \\ &\geq \Phi^{p-1}(x) - p\delta \varphi(\xi) \Phi^{p-2}(x) \\ &\geq \Phi^{p-1}(x) - \frac{p\delta}{\sqrt{2\pi}} \Phi^{p-2}(x). \end{aligned} \quad (3.15)$$

From this bound, we obtain

$$\int_{-\infty}^{\infty} \Phi^{p-1}(x - \delta) \varphi(x) dx \geq \frac{1}{p} - \frac{p\delta}{(p-1)\sqrt{2\pi}}. \quad (3.16)$$

Equations 3.13, 3.14 and 3.16 are then combined, which completes the proof. \square

Lemma 3.6. *Let X and Z be i.i.d. $N(0, 1)$ and $\varepsilon \in [0, 1]$. Then, for any $\delta > 0$,*

$$\Pr\left[\left|(1 - \sqrt{1 - \varepsilon})X - \sqrt{\varepsilon}Z\right| > \delta/2\right] \leq \frac{2}{\delta} \sqrt{\frac{\varepsilon}{\pi}}. \quad (3.17)$$

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

$$\sigma = \sqrt{(1 - \sqrt{1 - \varepsilon})^2 + \varepsilon} \leq \sqrt{2\varepsilon}. \quad (3.18)$$

Thus,

$$\Pr[|Y| > \delta/2] \leq 2\left(1 - \Phi(\delta/2\sigma)\right) \leq \frac{2\sigma}{\delta\sqrt{2\pi}} \leq \frac{2}{\delta} \sqrt{\frac{\varepsilon}{\pi}}. \quad (3.19)$$

\square

3.3 Positive results

In this section we construct an algorithm which approximates Max 2-Lin mod p within $p - \kappa(p)$, where $\kappa(p) > 0$, for all p . We do this by representing a variable in \mathbf{Z}_p by a constellation of vectors. The algorithm is constructed in several steps. First, we describe an algorithm which 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.

3.3.1 An algorithm for equations of the form $x_i - x_{i'} = c$

We first construct a semidefinite program which we can use to approximate the restriction of Max E2-Lin mod p where all equations are of the form $x_i - x_{i'} = c$, where $c \in \mathbf{Z}_p$. For each variable x_i we construct an object henceforth called a *porcupine* in the following way:

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

$$\langle v_j^i, v_k^i \rangle = \begin{cases} 1 & \text{when } j = k, \\ 0 & \text{otherwise,} \end{cases} \quad (3.20a)$$

for all i and all $j, k \in \mathbf{Z}_p$. We then add, for all pairs (i, i') , constraints ensuring that $\langle v_j^i, v_{j+k}^{i'} \rangle$ is independent of j for all k :

$$\langle v_j^i, v_{j+k}^{i'} \rangle = \langle v_{j'}^i, v_{j'+k}^{i'} \rangle \quad \text{for all } i, i' \text{ and all } j, j', k \in \mathbf{Z}_p. \quad (3.20b)$$

Finally, we introduce the constraints

$$\langle v_j^i, v_k^{i'} \rangle \geq 0 \quad \text{for all } i, i' \text{ and all } j, k \in \mathbf{Z}_p. \quad (3.20c)$$

We construct the objective function in the semidefinite program by including, for each equation $x_i - x_{i'} = c$, the following terms:

$$\frac{1}{p} \sum_{j=0}^{p-1} \langle v_j^i, v_{j+c}^{i'} \rangle. \quad (3.21)$$

The idea behind this is that two porcupines, corresponding to some equation, should be aligned in such a way that the corresponding terms in the objective function evaluate to one when the equation is satisfied.

Lemma 3.7. *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, 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 , such that x_i is assigned the value $\pi(x_i)$. Let $\{\hat{e}_j\}_{j=0}^{p-1}$ be orthonormal unit vectors in \mathbf{R}^p and set

$$v_{j-\pi(x_i)}^i = \hat{e}_j \quad \text{for all } i \text{ and all } j \in \mathbf{Z}_p. \quad (3.22)$$

The sum (3.21) corresponding to an equation $x_i - x_{i'} = c$ then takes the value

$$\frac{1}{p} \sum_{j=0}^{p-1} \langle v_j^i, v_{j+c}^{i'} \rangle = \frac{1}{p} \sum_{j=0}^{p-1} \langle \hat{e}_{j+\pi(x_i)}, \hat{e}_{j+c+\pi(x_{i'})} \rangle. \quad (3.23)$$

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

$$\frac{1}{p} \sum_{j=0}^{p-1} \langle \hat{e}_{j+\pi(x_i)}, \hat{e}_{j+c+\pi(x_{i'})} \rangle = 1. \quad (3.24)$$

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

$$\frac{1}{p} \sum_{j=0}^{p-1} \langle \hat{e}_{j+\pi(x_i)}, \hat{e}_{j+c+\pi(x_{i'})} \rangle = 0. \quad (3.25)$$

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

We can now formulate the algorithm to approximate Max E2-Lin mod p restricted to instances where every equation is 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 do the following:

1. Construct and solve the above semidefinite program.
2. If the optimum obtained from the semidefinite program is less than $1 - \kappa$ times the total weight of all equations, we select as our solution to the system of linear equations a random assignment to the variables x_i .
3. Otherwise, we 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 $N(0, 1)$ 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 now analyze this algorithm. Our aim is to show that it is possible to find, in probabilistic polynomial time, a solution approximating the optimum within $(1 - \kappa)p$. By Lemma 3.7 and step 2 in the algorithm, we will always approximate the optimum within $(1 - \kappa)p$ if the optimum of the semidefinite program is less than $1 - \kappa$ times the weight of all equations. Thus, we can assume in our analysis of the semidefinite algorithm 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 3.8. *If the optimum of the semidefinite program constructed above is at least $1 - \kappa$ times the weight of the instance, equations of total weight at least $1 - 2\kappa/\varepsilon$ times the weight of the instance have the property that the corresponding terms in the objective function evaluate to at least $\sqrt{1 - \varepsilon}$.*

Proof. Let μ be the fraction of the equations where the corresponding terms in the objective functions are less than $\sqrt{1 - \varepsilon}$. Then, the inequality

$$\mu\sqrt{1 - \varepsilon} + (1 - \mu) \geq 1 - \kappa \quad (3.26)$$

must always hold. When we solve for μ we obtain

$$\mu \leq \frac{\kappa}{1 - \sqrt{1 - \varepsilon}} \leq \frac{2\kappa}{\varepsilon}. \quad (3.27)$$

\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

$$\frac{1}{p} \sum_{j=0}^{p-1} \langle v_j^i, v_{j+c}^{i'} \rangle = \sqrt{1 - \varepsilon}, \quad (3.28)$$

where ε is small. By the constraints in Eq. 3.20b, Eq. 3.28 implies that

$$v_{j+c}^{i'} = \sqrt{1 - \varepsilon} v_j^i + \sqrt{\varepsilon} e_j, \quad (3.29)$$

where e_j is perpendicular to v_j^i and $\langle e_j, e_j \rangle = 1$. Thus, the probability that the two variables involved in the equation are assigned the wrong values should be small.

Definition 3.9. To simplify the notation, we introduce, for a fixed equation $x_i - x_{i'} = c$, the random variables

$$X_j = \langle v_j^i, r \rangle, \quad (3.30)$$

$$Y_j = \langle v_{j+c}^{i'}, r \rangle, \quad (3.31)$$

$$Z_j = \langle e_j, r \rangle. \quad (3.32)$$

By the construction of the porcupines and the choice of r , the X_i are i.i.d. $N(0, 1)$ and the Z_i are, possibly dependent, $N(0, 1)$. However, for each fixed i , X_i and Z_i are independent. To show that our algorithm has a performance ratio of at least $(1 - \kappa)p$, it is, by Lemma 3.8, sufficient to prove the following theorem:

Theorem 3.10. *It is possible to choose universal constants $\kappa < 1$ and $\varepsilon > 2\kappa$ such that, for all primes p ,*

$$\Pr[\text{equation satisfied}] > \frac{1}{p(1 - \kappa)(1 - 2\kappa/\varepsilon)} \quad (3.33)$$

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

Proof. The randomized rounding succeeds if the same j maximizes X_j and Y_j . Thus, 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 all other X_j are at most $(1 + \delta/2)\sqrt{2 \ln p}$. By Lemma 3.3,

$$\Pr[A(\delta)] > \frac{1}{2p^{2\delta + \delta^2}(1 + \delta)\sqrt{\pi \ln p}} \left(1 - \frac{1}{2 \ln p} - \frac{1}{2p^\delta \sqrt{\pi \ln p}} \right). \quad (3.34)$$

Next, let us study the Z_j . Let

$$B(\delta, \varepsilon) = \bigcap_{j=0}^{p-1} \left\{ |X_j - Y_j| < \frac{\delta}{4} \sqrt{(1 - \varepsilon) 2 \ln p} \right\}. \quad (3.35)$$

Since $Y_j = \sqrt{1 - \varepsilon}X_j + \sqrt{\varepsilon}Z_j$, we can use Lemma 3.4 to obtain

$$\Pr[\overline{B(\delta, \varepsilon)}] < \frac{4p^{1-\delta^2(1-\varepsilon)/32\varepsilon}}{\delta} \sqrt{\frac{2\varepsilon}{(1-\varepsilon)\pi \ln p}}. \quad (3.36)$$

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

$$\Pr[A(\delta) \cap B(\delta, \varepsilon)] \geq \Pr[A(\delta)] - \Pr[\overline{B(\delta, \varepsilon)}]. \quad (3.37)$$

It is now clear that we can choose constants δ , ε and κ which 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 theorem is valid also for $p < 23$. 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 3.5 and 3.6,

$$\Pr[C(\delta)] \geq 1 - \frac{p^2\delta}{(p-1)\sqrt{2\pi}}, \quad (3.38)$$

$$\Pr[\overline{D(\delta)}] \leq \frac{2p}{\delta} \sqrt{\frac{\varepsilon}{\pi}}. \quad (3.39)$$

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

$$\Pr[C(\delta) \cap D(\delta)] \geq \Pr[C(\delta)] - \Pr[\overline{D(\delta)}], \quad (3.40)$$

and a simple calculation shows that $\delta = 10^{-2}$, $\varepsilon = 10^{-7}$ and $\kappa = 10^{-8}$ work when $p \leq 23$. \square

Putting the pieces together, we obtain:

Theorem 3.11. *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 $(1 - \kappa)p$.*

Proof. The algorithm is as described above. If case 2 applies, then the naive randomized heuristic will approximate the solution within $(1 - \kappa)p$.

Otherwise, denote by w the total weight of the instance. By Lemma 3.8, 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 $\sqrt{1 - \varepsilon} \varepsilon$ in the optimal solution. By Theorem 3.10, 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 $w/p(1 - \kappa)$. \square

It is straightforward to adapt the algorithm to handle equations with one unknown. Simply introduce a new variable x_0 , representing 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.

Finally, since nothing in Sec 3.3.1 actually uses that p is prime, the results hold also for composite p .

3.3.2 General equations

In this section we extend the algorithm from Sec. 3.3.1 to handle general Max 2-Lin mod p instances. We do this by associating $p - 1$ porcupines, $\{v_j^{i,\ell}\}_{j=0}^{p-1}$ for $\ell \in \{1, 2, \dots, p-1\}$, 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 Sec. 3.3.1 with Eq. 3.20 generalized to

$$\langle v_j^{i,\ell}, v_k^{i,\ell} \rangle = \begin{cases} 1 & \text{when } j = k, \\ 0 & \text{otherwise,} \end{cases} \quad (3.41a)$$

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

$$\langle v_j^{i,\ell}, v_{j+k}^{i',\ell'} \rangle = \langle v_{j'}^{i,\ell}, v_{j'+k}^{i',\ell'} \rangle \quad (3.41b)$$

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

$$\langle v_j^{i,\ell}, v_k^{i',\ell'} \rangle \geq 0 \quad (3.41c)$$

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

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 condition 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 which approximates Max 2-Lin mod p within $p - \kappa(p)$, where $\kappa(p) > 0$ for all p , but goes towards zero as p grows to 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

$$\frac{1}{p(p-1)} \sum_{k=1}^{p-1} \sum_{j=0}^{p-1} \langle v_j^{i,ka}, v_{j+kc}^{i',kb} \rangle \quad (3.42)$$

in the objective function.

Lemma 3.12. *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 \mathbf{R}^p and set

$$v_{j-\ell\pi(x_i)}^{i,\ell} = \hat{e}_j \quad (3.43)$$

for all i , all $j \in \mathbf{Z}_p$ and all $\ell \in \mathbf{Z}_p^*$. The terms (3.42) corresponding to an equation $ax_i - bx_{i'} = c$ are then

$$\begin{aligned} & \frac{1}{p(p-1)} \sum_{k=1}^{p-1} \sum_{j=0}^{p-1} \langle v_j^{i,ka}, v_{j+kc}^{i',kb} \rangle \\ &= \frac{1}{p(p-1)} \sum_{k=1}^{p-1} \sum_{j=0}^{p-1} \langle \hat{e}_{j+ka\pi(x_i)}, \hat{e}_{j+kc+kb\pi(x_{i'})} \rangle \end{aligned} \quad (3.44)$$

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

$$\frac{1}{p(p-1)} \sum_{k=1}^{p-1} \sum_{j=0}^{p-1} \langle \hat{e}_{j+ka\pi(x_i)}, \hat{e}_{j+kc+kb\pi(x_{i'})} \rangle = 1. \quad (3.45)$$

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

$$\frac{1}{p(p-1)} \sum_{k=1}^{p-1} \sum_{j=0}^{p-1} \langle \hat{e}_{j+ka\pi(x_i)}, \hat{e}_{j+kc+kb\pi(x_{i'})} \rangle = 0. \quad (3.46)$$

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

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 the algorithm from Sec 3.3.1 as follows:

1. Construct and solve the above semidefinite program.
2. If the optimum obtained from the semidefinite program is less than $1 - \kappa(p)$ times the total weight of all equations, we select as our solution to the system of linear equations a random assignment to the variables x_i .
3. Otherwise, we 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 $N(0, 1)$ random variable. Then, we do the following:

Find the $j \in \mathbf{Z}_p$ maximizing $\langle v_j^{0,1}, r \rangle$.
 Set $t \leftarrow j$.
 For each $i \in \mathbf{Z}_p$,
 For all $j \in \mathbf{Z}_p$, set $q_{i,j} \leftarrow 0$.
 For all $k \in \mathbf{Z}_p^*$,
 Find the $j \in \mathbf{Z}_p$ 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 $j \in \mathbf{Z}_p$, set $q_{i,j} \leftarrow q_{i,j}/Q_i$.

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

Remark 3.13. By the choice of t in step 3 above, $q_{0,0}$ will always be non-zero. This turns out to be essential in the analysis.

By step 2 we approximate the solution within $(1 - \kappa)p$ if the semidefinite optimum is less than $1 - \kappa$ times the weight of all equations. We will now analyze the performance of step 3. We can assume that the semidefinite optimum is greater than $1 - \kappa$ times the total weight of the instance. By this assumption and Lemma 3.8, 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 (3.42) 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.,

$$\frac{1}{p(p-1)} \sum_{k=1}^{p-1} \sum_{j=0}^{p-1} \langle v_j^{i,ka}, v_{j+kc}^{i',kb} \rangle \geq \sqrt{1 - \varepsilon}. \quad (3.47)$$

We want to show that this equation is satisfied with large enough probability. Let us study the details of the selection procedure in step 3 above. Informally, our argument is:

- By the condition in Eq. 3.47 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

$$q_{i,a^{-1}j} \neq 0 \iff q_{i',b^{-1}(j+c)} \neq 0 \quad (3.48)$$

should hold for all j with high probability.

- If the above equivalence holds for all j , the randomized assignment at the end of step 3 above will find a satisfying assignment with a probability greater than $1/p$.

We now formalize this intuition.

Definition 3.14. For an arbitrary equation $ax_i - bx_{i'} = c$, let

$$X_j^k = \langle v_j^{i,ak}, r \rangle, Y_j^k = \langle v_{j+kc}^{i',bk}, r \rangle. \quad (3.49)$$

By the construction of the porcupines and the choice of r , the X_i^k are i.i.d. $N(0, 1)$.

Definition 3.15. Let ε_k be defined by the relation

$$\frac{1}{p} \sum_{j=0}^{p-1} \langle v_j^{i,ka}, v_{j+kc}^{i',kb} \rangle = \sqrt{1 - \varepsilon_k}. \quad (3.50)$$

By the constraints in Eq. 3.41, the above definitions imply that

$$Y_j^k = \sqrt{1 - \varepsilon_k} X_j^k + \sqrt{\varepsilon_k} Z_j^k, \quad (3.51)$$

where $Z_j^k \in N(0, 1)$.

Lemma 3.16. Let $ax_i - bx_{i'} = c$ be an arbitrary equation with the property that the corresponding terms in the objective function satisfy Eq. 3.47, and let A_k be the event that the same j maximizes X_j^k and Y_j^k . Then, for all $\delta > 0$,

$$\Pr[\overline{A_k}] \leq \frac{p^2 \delta}{(p-1)\sqrt{2\pi}} + \frac{2p}{\delta} \sqrt{\frac{\varepsilon_k}{\pi}}. \quad (3.52)$$

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

$$B_k(\delta) = \left\{ X_{(p)}^k > X_{(p-1)}^k + \delta \right\}, \quad (3.53)$$

$$C_k(\delta) = \bigcap_{i=0}^{p-1} \left\{ |X_i^k - Y_i^k| < \frac{\delta}{2} \right\}. \quad (3.54)$$

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,

$$B_k(\delta) \cap C_k(\delta) \subseteq A_k \implies \Pr[\overline{A_k}] \leq \Pr[\overline{B_k(\delta)}] + \Pr[\overline{C_k(\delta)}]. \quad (3.55)$$

By Lemma 3.5, we obtain the bound

$$\Pr[\overline{B_k(\delta)}] \leq \frac{p^2 \delta}{(p-1)\sqrt{2\pi}}, \quad (3.56)$$

and by Eq. 3.51 and Lemma 3.6, we obtain

$$\Pr[\overline{C_k(\delta)}] \leq \frac{2p}{\delta} \sqrt{\frac{\varepsilon_k}{\pi}}. \quad (3.57)$$

When Eqs. 3.55, 3.56, and 3.57 are combined, the proof follows. \square

Lemma 3.17. *For an arbitrary equation $ax_i - bx_{i'} = c$, 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 $j \in \mathbf{Z}_p$.*

Proof. 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

$$q_{i,a^{-1}k^{-1}(j'-t)} \neq 0 \iff q_{i',b^{-1}(k^{-1}(j'-t)+c)} \neq 0. \quad (3.58)$$

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

$$\sum_{j=0}^{p-1} q_{i,a^{-1}j} = \sum_{j=0}^{p-1} q_{i,j} = 1 \quad (3.59)$$

and

$$\sum_{j=0}^{p-1} q_{i',b^{-1}(j+c)} = \sum_{j=0}^{p-1} q_{i',j} = 1, \quad (3.60)$$

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

Lemma 3.18. *Let $ax_i - bx_{i'} = c$ be an arbitrary equation with the property that the corresponding terms in the objective function satisfy Eq. 3.47. Then,*

$$\Pr \left[\bigcap_{j \in \mathbf{Z}_p} q_{i,j} = q_{i',b^{-1}(aj+c)} \right] \geq 1 - \frac{p^2 \delta}{\sqrt{2\pi}} - \frac{2p(p-1)}{\delta} \sqrt{\frac{\varepsilon}{\pi}}, \quad (3.61)$$

where $\delta > 0$ is arbitrary.

Proof. By Lemma 3.17,

$$\Pr \left[\bigcap_{j=0}^{p-1} q_{i,a^{-1}j} = q_{i',b^{-1}(j+c)} \right] \geq \Pr \left[\bigcap_{k=1}^{p-1} A_k \right], \quad (3.62)$$

and by Lemma 3.16,

$$\Pr \left[\bigcap_{k=1}^{p-1} A_k \right] \geq 1 - \sum_{k=1}^{p-1} \Pr [\overline{A_k}] \geq 1 - \frac{p^2 \delta}{\sqrt{2\pi}} - \frac{2p}{\delta \sqrt{\pi}} \sum_{k=1}^{p-1} \sqrt{\varepsilon_k}. \quad (3.63)$$

We can trivially bound ε_k from above by $p\varepsilon$, which gives us a bound on $\sum \sqrt{\varepsilon_k}$ in terms of p and ε . We can, however, easily show a tighter bound. Since the function

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

$$\sqrt{1-\varepsilon} \leq \sum_{k=1}^{p-1} \frac{\sqrt{1-\varepsilon_k}}{p-1} \leq \sqrt{1 - \sum_{k=1}^{p-1} \frac{\varepsilon_k}{p-1}}, \quad (3.64)$$

where the first inequality follows from Eqs. 3.47 and 3.50, and the second from Jensen's inequality. Thus,

$$\sum_{k=1}^{p-1} \frac{\varepsilon_k}{p-1} \leq \varepsilon. \quad (3.65)$$

Using the Cauchy-Schwartz inequality, we can use Eq. 3.65 to obtain the bound

$$\sum_{k=1}^{p-1} \sqrt{\varepsilon_k} \leq \sqrt{(p-1) \sum_{k=1}^{p-1} \varepsilon_k} \leq (p-1)\sqrt{\varepsilon}. \quad (3.66)$$

When this is combined with Eqs. 3.62 and 3.63, the proof follows. \square

Lemma 3.19. *Suppose that $q_{0,0} > 0$. If, for some arbitrary equation $ax_i - bx_{i'} = c$, $q_{i,j} = q_{i',b^{-1}(aj+c)}$ for all $j \in \mathbf{Z}_p$, then this equation is 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,

$$\Pr[\text{equation satisfied}] = \sum_j q_{i,j} q_{i',b^{-1}(aj+c)} = \sum_j q_{i,j}^2, \quad (3.67)$$

where the second equality follows from the initial requirement in the formulation of the lemma. By step 3 in the algorithm, 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

$$\sum_j q_{i,j}^2 = m \times \frac{1}{m^2} \geq \frac{1}{p-1}, \quad (3.68)$$

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

$$\Pr[\text{equation satisfied}] = q_{i,-a^{-1}c} = q_{0,0}. \quad (3.69)$$

Since $q_{0,0} \neq 0$ we know, by step 3 in the algorithm, that $q_{0,0} \geq 1/(p-1)$, and the proof follows. \square

Theorem 3.20. *It is possible to choose $\kappa(p) > 0$ and $\varepsilon(p) > 0$ such that, for all primes p ,*

$$\Pr[\text{equation satisfied}] > \frac{1}{p(1-\kappa)(1-2\kappa/\varepsilon)} \quad (3.70)$$

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

$$p(1-\kappa)(1-2\kappa/\varepsilon) \times \Pr[\text{equation satisfied}] > 1. \quad (3.71)$$

It follows from step 3 in the algorithm, together with Lemmas 3.16, 3.17, 3.18 and 3.19, that

$$\Pr[\text{equation satisfied}] > \frac{1}{p-1} \left(1 - \frac{p^2\delta}{\sqrt{2\pi}} - \frac{2p(p-1)}{\delta} \sqrt{\frac{\varepsilon}{\pi}} \right) \quad (3.72)$$

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

$$\delta(p) = \frac{c_1\sqrt{2\pi}}{p^3}, \quad (3.73)$$

$$\varepsilon(p) = \frac{c_1^2 c_2^2 \pi^2}{2p^{10}(p-1)^2}, \quad (3.74)$$

$$\kappa(p) = \frac{c_1^2 c_2^2 c_3 \pi^2}{4p^{11}(p-1)^2}, \quad (3.75)$$

for some positive constants c_1 , c_2 and c_3 . When we use this ansatz in Eq. 3.72 we obtain

$$\begin{aligned} \Pr[\text{equation satisfied}] &> \frac{1}{p} \left(1 + \frac{1}{p} \right) \left(1 - \frac{c_1 + c_2}{p} \right) \\ &= \frac{1}{p} \left(1 + \frac{1 - c_1 - c_2}{p} - \frac{c_1 + c_2}{p^2} \right). \end{aligned} \quad (3.76)$$

Thus,

$$\begin{aligned} &p(1-\kappa)(1-2\kappa/\varepsilon) \times \Pr[\text{equation satisfied}] \\ &> \left(1 - \frac{c_3}{p} - \frac{c_1^2 c_2^2 c_3 \pi^2}{4p^{11}(p-1)^2} \left(1 - \frac{c_3}{p} \right) \right) \left(1 + \frac{1 - c_1 - c_2}{p} - \frac{c_1 + c_2}{p^2} \right) \\ &> \left(1 + \frac{1 - c_1 - c_2 - c_3}{p} - \frac{c_1 + c_2 + c_3}{p^2} - \left(1 + \frac{1}{p} \right) \frac{c_1^2 c_2^2 c_3 \pi^2}{4p^{11}(p-1)^2} \right). \end{aligned} \quad (3.77)$$

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

$$\Pr[\text{equation satisfied}] > \frac{1}{p(1-\kappa)(1-2\kappa/\varepsilon)} \quad (3.78)$$

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

As an immediate corollary, the main theorem follows. It is proved in exactly the same way as Theorem 3.11.

Theorem 3.21. *For all primes p , there exists a randomized polynomial time algorithm approximating Max 2-Lin mod p within $(1 - \kappa(p))p$, where $\kappa(p) > 0$ for all p .*

Proof. The algorithm is as described above. If case 2 applies, then the naive randomized heuristic will approximate the solution within $(1 - \kappa(p))p$.

Otherwise, denote by w the total weight of the instance. By Lemma 3.8, equations with total weight at least $(1 - 2\kappa(p)/\varepsilon(p))w$ have the property that the corresponding terms in the objective function of the semidefinite program evaluate to at least $\sqrt{1 - \varepsilon(p)}$ in the optimal solution. By Theorem 3.20, 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 $w/p(1 - \kappa(p))$. \square

If we use the explicit value of $\kappa(p)$ from the proof of Theorem 3.20, we see that Max 2-Lin mod p is approximable within $p - \Theta(p^{-12})$.

It is possible to generalize the algorithm to Max 2-Lin mod m for composite m . Let us first show that it is possible to approximate the optimum within m . Suppose that $m = p_1^{\alpha_1} \cdots p_k^{\alpha_k}$ and that the equations are of the form

$$ax_i - bx_{i'} = c. \quad (3.79)$$

Since equations where $\gcd(a, b, m)$ does not divide c can never be satisfied, we can remove them from the instance. It suffices to show that we can always satisfy a fraction $1/m$ of the remaining equations. To achieve this, we iterate over all primes p_s in the decomposition of m . The goal is to construct in each iteration the value of $x_i \bmod p_s^{\alpha_s}$ in such a way that a fraction $1/p_s^{\alpha_s}$ of the equations are satisfied mod $p_s^{\alpha_s}$, and then remove the unsatisfied equations from the instance. When this has been done for all p_s we will have an assignment to the x_i such that a fraction $1/m$ of the equations are satisfied mod m . We now show that it is enough to guess, in each iteration, the value of $x_i \bmod p_s^{\alpha_s}$ uniformly at random.

Lemma 3.22. *If we guess an assignment to the $x_i \bmod p_s^{\alpha_s}$ uniformly at random,*

$$\Pr[\text{equation satisfied}] \geq 1/p_s^{\alpha_s} \quad (3.80)$$

for an equation of the form

$$ax_i - bx_{i'} = c \bmod p_s^{\alpha_s}. \quad (3.81)$$

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

$$\frac{a}{p^t}x_i - \frac{b}{p^t}x_{i'} = \frac{c}{p^t} \bmod p_s^{\alpha_s-t}. \quad (3.82)$$

This equation will be satisfied with probability greater than $1/p_s^{\alpha_s}$. \square

To show that it is always possible to approximate the optimum within $(1 - \kappa(m))m$, only a slight modification of the above procedure is necessary: If the optimum is close to a fraction $1/m$ of all equations, there must be some prime factor p in m with the property that when we study the system mod p in the iterative procedure above, the semidefinite optimum is large, in the sense of Lemma 3.8. For this p , we use the rounded semidefinite solution to solve the system mod p . For all other factors in m we use the naive randomized heuristic described above. This will give a solution which approximates the optimum within $(1 - \kappa(p))m < (1 - \kappa(m))m$.

3.4 Negative results

In this section we show that there exists a universal constant, such that it is **NP**-hard to approximate Max E2-Lin mod p within that constant. To do this, we construct a gadget which reduces Max E3-Lin mod p to Max E2-Lin mod p . This gadget is valid for all primes $p \geq 3$. However, we cannot use it to show that it is **NP**-hard to approximate Max E2-Lin mod p within a constant factor. To deal with this, we combine the gadget with results by Andersson, Engebretsen and Håstad [4].

For the case $p = 2$, it is possible to use the methods of Trevisan et al. [28] to construct a gadget reducing Max E3-Lin mod 2 to Max E2-Lin mod 2 [27]. When this gadget is combined with the hardness results by Håstad [19], it follows that it is **NP**-hard to approximate Max E2-Lin mod 2 within $12/11 - \epsilon$. 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 although Trevisan et al. [28] have constructed an algorithm which computes optimal gadgets, we cannot use this algorithm to construct the gadgets for $p \geq 3$; the running time of the algorithm is simply too large.

We take an instance of Max E3-Lin mod p , where the coefficients in the equations are always one. For each equation in the instance we construct a number of equations with two variables per equation. We assume that the original equation is of the form

$$x_{i_1} + x_{i_2} + x_{i_3} = c. \quad (3.83)$$

We now study assignments to the variables x_i 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 $(x_{i_1}, x_{i_2}, x_{i_3}) = (0, a_{i,j}, b_{i,j})$ we introduce a new auxiliary variable $y_{i,j}$ and construct the following group of equations:

$$x_{i_1} + y_{i,j} = c, \quad (3.84a)$$

$$x_{i_2} + y_{i,j} = a_{i,j}, \quad (3.84b)$$

$$x_{i_3} + (p-2)y_{i,j} = b_{i,j}. \quad (3.84c)$$

Lemma 3.23. *When $p \geq 3$ is prime, the above construction is a $((p-1)(p+3), 1)$ -gadget.*

Proof. Let π be an assignment to the x_i and the $y_{i,j}$. We will study some arbitrary equation (3.83) from the Max E3-Lin mod p instance, and the corresponding $p^2 - p$ groups (3.84) from the Max E2-Lin mod p instance.

Assume that the assignment π satisfies Eq. 3.83. Then, there is no assignment to $y_{i,j}$, such that all equations in the groups (3.84) are satisfied. For each j we can, however, always satisfy one of the three equations containing $y_{i,j}$ by choosing $\pi(y_{i,j}) = -c$. In fact, it may be possible to satisfy two of the three equations. Specifically, this is possible if $(c - \pi(y_{i,j}), a_{i,j} - \pi(y_{i,j}), b_{i,j} - (p-2)\pi(y_{i,j}))$ and $(\pi(x_{i_1}), \pi(x_{i_2}), \pi(x_{i_3}))$ differ in only one position for some assignment to $y_{i,j}$.

Exactly $3(p-1)$ of the groups (3.84) have this property. For, suppose that the satisfying assignment is (s_1, s_2, s_3) . Then, there are exactly $3(p-1)$ ways to construct unsatisfying assignments $(u_{1,j}, u_{2,j}, u_{3,j})$ 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. Each such assignment corresponds to the group $(0, a_{i,j}, b_{i,j})$ where $a_{i,j} = u_{2,j} - u_{1,j}$ and $b_{i,j} = u_{3,j} - (p-2)u_{2,j}$. With the choice $\pi(y_{i,j}) = c - u_{1,j}$, two of the three equations in the group will be satisfied. Finally, two different unsatisfying assignments $(u_{1,j}, u_{2,j}, u_{3,j})$ and $(u_{1,k}, u_{2,k}, u_{3,k})$, both with the property that they differ from the satisfying assignment in exactly one position, can never correspond to the same group. For, if that were the case, the equations

$$u_{2,j} - u_{1,j} = u_{2,k} - u_{1,k} \quad (3.85)$$

$$u_{3,j} - (p-2)u_{1,j} = u_{3,k} - (p-2)u_{1,k} \quad (3.86)$$

$$u_{\ell,j} = u_{\ell,k} \quad \text{for some } \ell \in \{1, 2, 3\} \quad (3.87)$$

would have to be simultaneously satisfied. This, however, implies that $u_{\ell,j} = u_{\ell,k}$ for all ℓ .

Summing up, the contribution to the objective function is

$$2 \times 3(p-1) + (p^2 - p) - 3(p-1) = (p-1)(p+3). \quad (3.88)$$

Let us now assume that the assignment π does not satisfy Eq. 3.83. Then for some j , all three equations containing $y_{i,j}$ can be satisfied. By a similar argument as above, there exists $3(p-2)$ groups (3.84) where two equations can be satisfied

and in the remaining groups one equation can be satisfied. The contribution to the objective function is

$$3 + 2 \times 3(p - 2) + (p^2 - p) - (3(p - 2) + 1) = (p - 1)(p + 3) - 1. \quad (3.89)$$

Thus, the construction is a $((p - 1)(p + 3), 1)$ -gadget. \square

We can now combine this gadget with a suitable hardness result for Max E3-Lin mod p to show a hardness result for Max E2-Lin mod p .

Theorem 3.24. *For all $\varepsilon > 0$ and all $p \geq 3$, it is **NP**-hard to approximate Max E2-Lin mod p within $(p^2 + 3p)/(p^2 + 3p - 1) - \varepsilon$.*

Proof. By the result of Håstad [19], it is, for any $\varepsilon > 0$, **NP**-hard to approximate Max E3-Lin mod p within $p - \varepsilon$ also in the special case when all coefficients in the equations are equal to one. When this result is combined with Lemma 3.23, the theorem follows. \square

It has been shown by Andersson, Engebretsen and Håstad [4] that for $p \geq 11$ it is **NP**-hard to approximate Max E2-Lin mod p within $18/17 - \varepsilon$ for all $\varepsilon > 0$. When this result is combined with Theorem 3.24 we obtain the following general result:

Theorem 3.25. *For every prime p , it is **NP**-hard to approximate Max E2-Lin mod p within $70/69 - \varepsilon$.*

Proof. For $p = 2$ we use Sorkin's gadget [27] combined with Håstad's hardness result [19]. For $p \in \{3, 5, 7\}$ we use Theorem 3.24, and for $p \geq 11$ we use the result of Andersson, Engebretsen and Håstad [4]. \square

It is easy to see, that if p is a prime factor in m , Max Ek -Lin mod p is actually a special case of Max Ek -Lin mod m . For, if we have a Max Ek -Lin mod p instance, we can convert it into an Max Ek -Lin mod m instance by multiplying each equation with m/p .

3.5 Conclusions

We have shown that it is **NP**-hard to approximate Max E2-Lin mod p within $70/69 - \varepsilon$, independently of p , and that there exists a randomized polynomial time algorithm approximating Max 2-Lin mod p within $p - \Theta(p^{-12})$. For the special case of Max 2-Lin mod p , where the equations are either of the form $x_i - x_{i'} = c$ or $x_i = c$, we have shown that there exists a randomized polynomial time algorithm approximating the problem within $(1 - 10^{-8})p$. Of major interest at this point is, in our opinion, to determine if the lower bound is in fact increasing with p , or if there exists a polynomial time algorithm approximating Max 2-Lin mod p within some constant ratio.

The algorithm proposed in Sec. 3.3.2 can never approximate the solution within a factor better than $p - 1$, by the construction of the randomized rounding. In fact, even for a satisfiable system of linear equations, the expected weight of the equations satisfied by the algorithm is a fraction $1/(p - 1)$ of the total weight of the instance. This behavior is not desired, and indicates that it should be possible to construct a better algorithm.

Chapter 4

Denseness improves approximability

In this chapter, we study dense instances of systems of linear equations modulo some prime p . Specifically, we study systems with exactly k unknowns in each equation and $\Theta(n^k)$ equations, where n is the number of variables. We show that there exists a randomized polynomial time approximation scheme for such instances.

4.1 Preliminaries

Definition 4.1. We denote by $\text{Max } Ek\text{-Lin mod } p$ 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.

Definition 4.2. We denote by $\text{Max } k\text{-Function Sat}$ the problem in which the input consists of a number of boolean functions in n boolean variables. Each function depends on k variables. The objective is to find the assignment maximizing the number of satisfied functions.

Definition 4.3. The class **Max-SNP** is the class of optimization problems which can be written in the form

$$\max_S |\{x : \Phi(I, S, x)\}|, \quad (4.1)$$

where Φ is a quantifier-free formula, I an instance and S a solution.

Arora, Karger and Karpinski [6] 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 $\varepsilon > 0$. The methods do not seem to generalize to linear equations mod p since the integer programs needed to express such an instance do not have the required properties.

Fernandez de la Vega [11] has also constructed a randomized polynomial time approximation scheme for dense instances of Max Cut, independently of Arora, Karger and Karpinski. However, it seems difficult to generalize his construction to other problems.

In this chapter, we use methods and ideas introduced by Goldreich, Goldwasser and Ron [18]. In their randomized polynomial time approximation scheme for Max Cut, they partition the vertices of the graph 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.

Frieze and Kannan [13] claim that they have constructed an algorithm giving 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 an instance of Max k -Function Sat with $\Theta(n^k)$ functions. It is not immediately clear that a dense instance of linear equations mod p can be described in this manner, and on top of that, the algorithm proposed in this chapter has a simpler structure and shorter running time than their algorithm.

4.2 Systems with two unknowns per equation

We consider an unweighted system of linear equations modulo some prime p . There are in total n different variables in the system. The equations are of the form

$$ax_i + bx_{i'} = c \tag{4.2}$$

where $i \neq i'$, $a, b \in \mathbf{Z}_p^*$, and $c \in \mathbf{Z}_p$. We assume that there are no equivalent equations in the system. I.e., if the two equations

$$ax_i + bx_{i'} = c \tag{4.3}$$

$$a'x_i + b'x_{i'} = c' \tag{4.4}$$

are both in the system, we assume that there is no $d \in \mathbf{Z}_p$, such that $a = da'$, $b = db'$ and $c = dc'$. We think of variable assignments as functions from the set of unknowns to \mathbf{Z}_p .

Definition 4.4. Denote by $S(X, \tau, x \leftarrow r)$ the number of satisfied equations with one unknown from the set X and x as the other unknown, given that the variables in X are assigned values according to the function τ and x is assigned the value r .

Definition 4.5. If $\pi_1: V_1 \mapsto \mathbf{Z}_p$ and $\pi_2: V_2 \mapsto \mathbf{Z}_p$ are two assignments, and $V_1 \cap V_2 = \emptyset$, we denote by $\pi_1 \otimes \pi_2$ the assignment π such that

$$\pi(v) = \begin{cases} \pi_1(v) & \text{if } v \in V_1, \\ \pi_2(v) & \text{if } v \in V_2. \end{cases} \quad (4.5)$$

The algorithm we use is based on the Max Cut algorithm by Goldreich, Goldwasser and Ron [18], and we use their terminology and notation.

Algorithm 4.6. Approximation algorithm for dense instances of Max E2-Lin mod p :

1. Divide the variable set V into ℓ disjoint sets $\{V^i\}_{i=1}^\ell$, each of size n/ℓ .
2. Choose ℓ sets $\{U^i\}_{i=1}^\ell$ in the following way: 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 $\pi: U \mapsto \mathbf{Z}_p$, form an assignment $\Pi_\pi: V \mapsto \mathbf{Z}_p$ in the following way:
 - (a) For $i \in \{1, \dots, \ell\}$, an assignment $\Pi_\pi^i: V^i \mapsto \mathbf{Z}_p$ is constructed as follows: For each $v \in V^i$, let $j^*(v)$ be the $j \in \mathbf{Z}_p$ which maximizes $S(U^i, \pi, v \leftarrow j)$. Then define $\Pi_\pi^i(v) = j^*(v)$.
 - (b) Merge the above assignments in the obvious way:

$$\Pi_\pi = \bigotimes_{i=1}^\ell \Pi_\pi^i. \quad (4.6)$$

4. Let Π be the variable assignment Π_π which maximizes the number of satisfied equations.
5. Return Π .

The running time of the algorithm is $O(n^2)$. The parameters ℓ and t are independent of n . They will be determined during the analysis of the algorithm.

Our overall goal is to show that the algorithm 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 , ε_2 and ε_3 . They are all linear in ε , and determined later.

The intuition behind the algorithm is as follows: Since the graph 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 H^i 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 . Let us formalize this.

Definition 4.7. Let H^i be an assignment to the variables in $V \setminus V^i$. We say that the set U^i is *good* with respect to $V \setminus V^i$ if for all except a fraction of at most ε_1 of the variables $v \in V^i$ we have that

$$\left| \frac{S(U^i, H^i|_{U^i}, v \leftarrow j)}{t} - \frac{S(V \setminus V^i, H^i, v \leftarrow j)}{n - |V^i|} \right| \leq \varepsilon_2 \quad \text{for all } j \in \mathbf{Z}_p. \quad (4.7)$$

Lemma 4.8. *For a fixed i , 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 $1 - \delta/\ell$.*

Proof. Fix a variable $v \in V^i$ and some $j \in \mathbf{Z}_p$. Note that the assignment H^i is fixed; the underlying probability space is the possible choices of U^i . In other words,

$$\Omega = \{\omega \subseteq V \setminus V^i : |\omega| = t\} \quad (4.8)$$

and U^i is chosen uniformly from Ω . We now introduce, for each $w \in V \setminus V^i$ a Bernoulli random variable $\xi_{i,j,v,w}$ with the property that

$$\xi_{i,j,v,w} = \begin{cases} 1 & \text{if } w \in U^i, \\ 0 & \text{otherwise.} \end{cases} \quad (4.9)$$

We can use these random variables to express the number of satisfied equations containing v and one variable from U^i .

$$S(U^i, H^i|_{U^i}, v \leftarrow j) = \sum_{w \in V \setminus V^i} S(\{w\}, H^i|_{\{w\}}, v \leftarrow j) \xi_{i,j,v,w}. \quad (4.10)$$

Since U^i is chosen uniformly at random from $V \setminus V^i$,

$$\Pr[\xi_{i,j,v,w} = 1] = \frac{t}{n - |V^i|}. \quad (4.11)$$

Now we construct the random variable

$$X_{i,j,v} = \frac{S(U^i, H^i|_{U^i}, v \leftarrow j)}{t}. \quad (4.12)$$

From Eqs. 4.10 and 4.11 it follows that

$$E[X_{i,j,v}] = \sum_{w \in V \setminus V^i} \frac{S(\{w\}, H^i|_{\{w\}}, v \leftarrow j)}{n - |V^i|} = \frac{S(V \setminus V^i, H, v \leftarrow j)}{n - |V^i|}, \quad (4.13)$$

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^i and the instance. Since there is, for each $w \in V \setminus V^i$, at most $p(p-1)$ equations containing v and w ,

$S(\{w\}, H^i|_{\{w\}}, v \leftarrow j)$ can be anywhere between 0 and $p - 1$, which complicates things even worse. Fortunately, we can use martingale tail bounds to reach our goal in spite of our limited knowledge of the probability distribution of $X_{i,j,v}$. Let $\mathcal{F}_0 = \{\emptyset, \Omega\}$. For $m > 0$ we partition Ω into disjoint subsets, such that all t -tuples in a subset have their first m coordinates in common. Then, we let \mathcal{F}_m be the σ -field generated by that partition. We stop this process with $\mathcal{F}_t = 2^\Omega$. The sequence $\{\mathcal{F}_m\}_{m=0}^t$ is a filtration, and thus the sequence $\{X_{i,j,v}^m\}_{m=0}^t$, where

$$X_{i,j,v}^m = \mathbb{E}[X_{i,j,v} \mid \mathcal{F}_m], \quad (4.14)$$

is a martingale [24]. We note that this definition implies that

$$|X_{i,j,v} - \mathbb{E}[X_{i,j,v}]| = |X_{i,j,v}^t - X_{i,j,v}^0|. \quad (4.15)$$

Furthermore we observe that

$$|X_{i,j,v}^m - X_{i,j,v}^{m-1}| \leq (p-1)/t \quad \text{for all } m \in \{1, \dots, t\}. \quad (4.16)$$

For, when we expose the m th variable, the expected number of satisfied equations, conditioned on the exposed variables, can change by at most $p - 1$. This enables us to apply Azuma's inequality [24]:

$$\Pr[|X_{i,j,v} - \mathbb{E}[X_{i,j,v}]| > \varepsilon_2] < 2e^{-\varepsilon_2^2 t / 2(p-1)^2}. \quad (4.17)$$

The above inequality is valid for fixed i, j and v . A set U^i is good if the above inequality holds for all j and for all but ε_1 of the vertices in V^i . Thus we keep i and j fixed and construct a new family of Bernoulli random variables

$$\eta_{i,j,v} = \begin{cases} 1 & \text{if } |X_{i,j,v} - \mathbb{E}[X_{i,j,v}]| > \varepsilon_2, \\ 0 & \text{otherwise,} \end{cases} \quad (4.18)$$

and set

$$Y_{i,j} = \frac{1}{|V^i|} \sum_{v \in V^i} \eta_{i,j,v}. \quad (4.19)$$

By Eq 4.17,

$$\Pr[\eta_{i,j,v} = 1] < 2e^{-\varepsilon_2^2 t / 2(p-1)^2}, \quad (4.20)$$

and thus

$$\mathbb{E}[Y_{i,j}] < 2e^{-\varepsilon_2^2 t / 2(p-1)^2}. \quad (4.21)$$

We can now use Markov's inequality to bound

$$\Pr[Y_{i,j} > \varepsilon_1] \leq \frac{\mathbb{E}[Y_{i,j}]}{\varepsilon_1} < \frac{2e^{-\varepsilon_2^2 t / 2(p-1)^2}}{\varepsilon_1}. \quad (4.22)$$

The above inequality is valid for a fixed j , and it can be used to obtain the probability that U^i is good:

$$\Pr[U^i \text{ is good}] = \Pr \left[\bigcap_{j=1}^p Y_{i,j} \leq \varepsilon_1 \right] \geq 1 - \frac{2pe^{-\varepsilon_2^2 t / 2(p-1)^2}}{\varepsilon_1}. \quad (4.23)$$

Finally, we are to determine a suitable value for t , in order to make this probability large enough. If we choose

$$t \geq \frac{2(p-1)^2}{\varepsilon_2^2} \ln \frac{2\ell p}{\delta \varepsilon_1}, \quad (4.24)$$

the probability that U^i is good will be at least $1 - \delta/\ell$. \square

Corollary 4.9. *The probability that all U^i are good is at least $1 - \delta$.*

Proof. There are ℓ different U^i . \square

For any assignment H to the variables, we can use the assignment $H^i = H|_{V \setminus V^i}$ and step 3a in the algorithm to construct an assignment Π^i to the variables in V^i . If U^i is good, we expect the number of equations satisfied by $H^i \otimes \Pi^i$ to be close to the number of equations satisfied by H . To formalize this intuition, we need the following definitions.

Definition 4.10.

$$\mu(\tau) = \frac{\# \text{ equations satisfied by the assignment } \tau}{n^2}. \quad (4.25)$$

Definition 4.11. We say that a variable $v \in V^i$ is *unbalanced* with respect to $V \setminus V^i$ if there exists a $j^* \in \mathbf{Z}_p$ such that for all $j' \in \mathbf{Z}_p \setminus \{j^*\}$ we have that

$$\frac{S(V \setminus V^i, H, v \leftarrow j^*)}{n - |V^i|} \geq \frac{S(V \setminus V^i, H, v \leftarrow j')}{n - |V^i|} + \varepsilon_3. \quad (4.26)$$

Lemma 4.12. *Let $H: V \mapsto \mathbf{Z}_p$ be a given assignment and δ and ε be fixed, positive constants. Let $\pi = H|_U$ and Π^i be the assignment produced as in step 3a in the algorithm. Then, if U^i is good, it is possible to choose the constant ℓ in such a way that*

$$\mu(H|_{V \setminus V^i} \otimes \Pi^i) \geq \mu(H) - \varepsilon/p\ell. \quad (4.27)$$

Proof. We want to compare the number of equations satisfied by the assignment H to the number satisfied by $H|_{V \setminus V^i} \otimes \Pi^i$. 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 four 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 $(p-1)n^2/2\ell^2$ 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 $v \in V^i$ is unbalanced and satisfies Eq. 4.7, and $w \notin V^i$. If we combine Eq. 4.7 and Eq. 4.26 we obtain the inequality

$$\frac{S(U^i, \pi, v \leftarrow j^*)}{t} \geq \frac{S(U^i, \pi, v \leftarrow j')}{t} + \varepsilon_3 - 2\varepsilon_2 \quad \text{for all } j'. \quad (4.28)$$

By the construction of the algorithm, the value chosen for v will thus be the correct value, provided that $\varepsilon_3 > 2\varepsilon_2$. It follows that the number of satisfied equations of this type cannot decrease.

3. Equations of the form $av + bw = c$ where $v \in V^i$ is *not* unbalanced, but still satisfies Eq. 4.7, and $w \notin V^i$. In this case, the algorithm may select the wrong assignment to v . However, that cannot decrease the optimum value 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 Eq. 4.7, this implies that

$$\left| \frac{S(V \setminus V^i, H, v \leftarrow j')}{n - |V^i|} - \frac{S(V \setminus V^i, H, v \leftarrow j)}{n - |V^i|} \right| \leq 2\varepsilon_2 < \varepsilon_3. \quad (4.29)$$

Since there are at most $|V^i|$ different v that are not unbalanced, we can bound the decrease in the number of satisfied equations by

$$|V^i|(S(V \setminus V^i, H, v \leftarrow j') - S(V \setminus V^i, H, v \leftarrow j)) \leq 2\varepsilon_2 n^2 / \ell. \quad (4.30)$$

4. Equations of the form $av + bw = c$ where $v \in V^i$ does not satisfy Eq. 4.7, 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 $\varepsilon_1 p^2 |V^i| n / 2$. Only $\varepsilon_1 p |V^i| n / 2$ of these can be satisfied at the same time, and hence a bound on the decrease is $\varepsilon_1 p |V^i| n / 2 = \varepsilon_1 p n^2 / 2\ell$.

Summing up, the total decrease is at most

$$pn^2/2\ell^2 + 2\varepsilon_2 n^2/\ell + \varepsilon_1 pn^2/2\ell. \quad (4.31)$$

If we select $\ell = p^2/\varepsilon$, $\varepsilon_1 = \varepsilon/2p$, $\varepsilon_2 = \varepsilon/8$, and $\varepsilon_3 = \varepsilon/3$, the total decrease is at most

$$\varepsilon n^2/2p\ell + \varepsilon n^2/4p\ell + \varepsilon n^2/4p\ell = \varepsilon n^2/p\ell, \quad (4.32)$$

which concludes the proof. \square

Corollary 4.13. *If we construct from an assignment $\pi = H|_U$, a new assignment Π_π as in step 3 of Algorithm 4.6, this new assignment has the property that $\mu(\Pi_\pi) \geq \mu(H) - \varepsilon/p$.*

Proof. Apply Lemma 4.12 ℓ times, one for each $i \in \{1, \dots, \ell\}$. \square

The only observation needed now is, that since Corollary 4.13 is valid for any assignment H , we can obtain a randomized polynomial time approximation scheme by considering the case when H is the optimal assignment.

Theorem 4.14. *For instances of Max E2-Lin mod p where the number of equations is $\Theta(n^2)$, Algorithm 4.6 is a randomized polynomial time approximation scheme.*

Proof. 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 4.9 and 4.13 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 an Max E2-Lin mod p instance with cn^2 equations is at least cn^2/p , this gives a relative error which is at most ε/c . \square

4.3 Systems with k unknowns per equation

It turns out that the methods used in Sec. 4.2 generalize to the case with k unknowns per equation, if we make suitable changes to the definitions used. For the sake of completeness, we repeat the argument once again, with the details worked out. We consider systems of linear equations modulo some prime p of the form

$$\mathbf{a} \cdot \mathbf{x} = c \tag{4.33}$$

where $\mathbf{x} \in V^k$, $\mathbf{a} \in (\mathbf{Z}_p^*)^k$, and $c \in \mathbf{Z}_p$. We assume that there are no equivalent equations in the system. We can use exactly the same algorithm as above if we change the definition of $S(X, \tau, x \leftarrow r)$ as follows:

Definition 4.15. We extend the notation

$$S(X, \tau, x \leftarrow r) \tag{4.34}$$

to mean the number of satisfied equations with $k - 1$ unknowns from the set X and one unknown $x \notin X$, given that the variables in X are assigned values according to the function τ and x is assigned the value r .

Definition 4.16. Let H^i be an assignment to the variables in $V \setminus V^i$. We say that the set U^i is *good* with respect to $V \setminus V^i$ if for all except a fraction of at most ε_1 of the variables $v \in V^i$ we have that

$$\left| \frac{S(U^i, \pi, v \leftarrow j)}{\binom{t}{k-1}} - \frac{S(V \setminus V^i, H, v \leftarrow j)}{\binom{n-|V^i|}{k-1}} \right| \leq \varepsilon_2 \quad \text{for all } j \in \mathbf{Z}_p. \tag{4.35}$$

Lemma 4.17. *For a fixed i , 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 $1 - \delta/\ell$.*

Proof. Fix a variable $v \in V^i$ and some $j \in \mathbf{Z}_p$. It turns out that the analysis is easier if the sample space Ω is represented somewhat differently. Specifically,

$$\Psi = \{\psi \subseteq V \setminus V^i : |\psi| = t\}, \quad (4.36)$$

$$\Omega = \{\omega : (\exists \psi \in \Psi : s \in \omega \implies (s \subseteq \psi \wedge |s| = k-1))\}. \quad (4.37)$$

To choose U^i we choose an ω uniformly at random from Ω and set

$$U^i = \bigcup_{u_m \in \omega} u_m. \quad (4.38)$$

This in fact chooses U^i uniformly at random from $V \setminus V^i$. We now introduce, for each $\mathbf{w} \subseteq V \setminus V^i$ such that $|\mathbf{w}| = k-1$, a Bernoulli random variable $\xi_{i,j,v,\mathbf{w}}$ with the property that

$$\xi_{i,j,v,\mathbf{w}} = \begin{cases} 1 & \text{if } \mathbf{w} \subseteq U^i, \\ 0 & \text{otherwise.} \end{cases} \quad (4.39)$$

We can use these random variables to express the number of satisfied equations containing v and $k-1$ variables from U^i .

$$S(U^i, H^i |_{U^i}, v \leftarrow j) = \sum_{\substack{\mathbf{w} \subseteq V \setminus V^i \\ |\mathbf{w}| = k-1}} S(\mathbf{w}, H^i |_{\mathbf{w}}, v \leftarrow j) \xi_{i,j,v,\mathbf{w}}. \quad (4.40)$$

Since U^i is chosen uniformly at random from $V \setminus V^i$,

$$\Pr[\xi_{i,j,v,\mathbf{w}} = 1] = \frac{\binom{n-|V^i|-k+1}{t-k+1}}{\binom{n-|V^i|}{t}} = \frac{\binom{t}{k-1}}{\binom{n-|V^i|}{k-1}}. \quad (4.41)$$

Now we construct the random variable

$$X_{i,j,v} = \frac{S(U^i, H^i |_{U^i}, v \leftarrow j)}{\binom{t}{k-1}}. \quad (4.42)$$

From Eqs. 4.40 and 4.41 it follows that

$$E[X_{i,j,v}] = \frac{S(V \setminus V^i, H, v \leftarrow j)}{\binom{n-|V^i|}{k-1}}. \quad (4.43)$$

As in the proof of Lemma 4.8, we want to bound

$$\Pr[|X_{i,j,v} - E[X_{i,j,v}]| > \varepsilon_2] \quad (4.44)$$

using martingale tail bounds. To reach this goal we construct, for fixed i, j and v , a $(k-1)$ -tuple exposure martingale. To simplify the notation, we put

$$T = \binom{t}{k-1}. \quad (4.45)$$

Note that each $\omega \in \Omega$ consists of exactly T tuples. Let $\mathcal{F}_0 = \{\emptyset, \Omega\}$. For $m > 0$ we partition Ω into disjoint subsets, such that all members of a particular subset have their first m tuples in common. Then, we let \mathcal{F}_m be the σ -field generated by that partition. We stop this process with $\mathcal{F}_T = 2^\Omega$. The sequence $\{\mathcal{F}_m\}_{m=0}^T$ is a filtration, and thus the sequence $\{X_{i,j,v}^m\}_{m=0}^T$, where

$$X_{i,j,v}^m = \mathbb{E}[X_{i,j,v} \mid \mathcal{F}_m], \quad (4.46)$$

is a martingale [24]. The Lipschitz condition in Eq. 4.16 changes to

$$|X_{i,j,v}^m - X_{i,j,v}^{m-1}| \leq (p-1)^{k-1}/T \quad \text{for all } m \in \{1, \dots, T\} \quad (4.47)$$

since, for an $u \in U^i$ and a $(k-1)$ -tuple $\mathbf{w} \subseteq V \setminus V^i$, at most $(p-1)^{k-1}$ of the equations containing u and \mathbf{w} can be satisfied at the same time. This enables us to apply Azuma's inequality [24]:

$$\Pr[|X_{i,j,v} - \mathbb{E}[X_{i,j,v}]| > \varepsilon_2] < 2e^{-\varepsilon_2^2 T / 2(p-1)^{2(k-1)}}. \quad (4.48)$$

The above inequality is valid for fixed $\{i, j, v\}$. A set U^i is good if the above inequality holds for all j and for all but ε_1 of the vertices in V^i . Thus we keep i and j fixed and construct a new family of Bernoulli random variables

$$\eta_{i,j,v} = \begin{cases} 1 & \text{if } |X_{i,j,v} - \mathbb{E}[X_{i,j,v}]| > \varepsilon_2, \\ 0 & \text{otherwise,} \end{cases} \quad (4.49)$$

and set

$$Y_{i,j} = \frac{1}{|V^i|} \sum_{v \in V^i} \eta_{i,j,v}. \quad (4.50)$$

By Eq 4.48,

$$\Pr[\eta_{i,j,v} = 1] < 2e^{-\varepsilon_2^2 T / 2(p-1)^{2(k-1)}}, \quad (4.51)$$

and thus

$$\mathbb{E}[Y_{i,j}] < 2e^{-\varepsilon_2^2 T / 2(p-1)^{2(k-1)}}. \quad (4.52)$$

We can now use Markov's inequality to bound

$$\Pr[Y_{i,j} > \varepsilon_1] \leq \frac{\mathbb{E}[Y_{i,j}]}{\varepsilon_1} < \frac{2e^{-\varepsilon_2^2 T / 2(p-1)^{2(k-1)}}}{\varepsilon_1}. \quad (4.53)$$

The above inequality is valid for a fixed j , and it can be used to obtain the probability that U^i is good:

$$\Pr[U^i \text{ is good}] = \Pr \left[\bigcap_{j=1}^p Y_{i,j} \leq \varepsilon_1 \right] \geq 1 - \frac{2pe^{-\varepsilon_2^2 T/2(p-1)^{2(k-1)}}}{\varepsilon_1}. \quad (4.54)$$

Finally, we are to determine a suitable value t , in order to make this probability large enough. If we choose

$$T \geq \frac{2(p-1)^{2(k-1)}}{\varepsilon_2^2} \ln \frac{2\ell p}{\delta \varepsilon_1}, \quad (4.55)$$

the probability that U^i is good will be at least $1 - \delta/\ell$. Since

$$T > (t - k + 2)^{k-1} / (k - 1)!, \quad (4.56)$$

we are safe if we choose

$$t \geq k - 2 + (p - 1)^2 \left(\frac{2(k - 1)!}{\varepsilon_2^2} \ln \frac{2\ell p}{\delta \varepsilon_1} \right)^{\frac{1}{k-1}}. \quad (4.57)$$

□

Corollary 4.18. *The probability that all U^i are good is at least $1 - \delta$.*

Proof. There are ℓ different U^i . □

Next, we extend the definitions of μ and unbalancedness:

Definition 4.19.

$$\mu(\tau) = \frac{\# \text{ equations satisfied by the assignment } \tau}{n^k}. \quad (4.58)$$

Definition 4.20. We say that a variable $v \in V^i$ is *unbalanced* with respect to $V \setminus V^i$ if there exists a $j^* \in \mathbf{Z}_p$ such that for all $j' \in \mathbf{Z}_p \setminus \{j^*\}$ we have that

$$\frac{S(V \setminus V^i, H, v \leftarrow j^*)}{\binom{n-|V^i|}{k-1}} \geq \frac{S(V \setminus V^i, H, v \leftarrow j')}{\binom{n-|V^i|}{k-1}} + \varepsilon_3. \quad (4.59)$$

Lemma 4.21. *Let $H: V \mapsto \mathbf{Z}_p$ be a given assignment and δ and ε be fixed, positive constants. Let $\pi = H|_U$ and Π^i be the assignment produced as in step 3a in the algorithm. Then, if U^i is good, it is possible to choose the constant ℓ in such a way that*

$$\mu(H|_{V \setminus V^i} \otimes \Pi^i) \geq \mu(H) - \varepsilon/p\ell. \quad (4.60)$$

Proof. To bound the decrease in the number of satisfied equations, we perform a case analysis similar to that in the proof of Lemma 4.12:

Equations with more than one variable from V^i . There are at most

$$p(p-1)^{k-1}(n/\ell)^2 n^{k-2}/k! \quad (4.61)$$

such equations, and less than

$$\frac{p(n/\ell)^2 n^{k-2}}{k!} = \frac{pn^k}{k!\ell^2} \quad (4.62)$$

can be satisfied by any given assignment, which bounds the decrease in the number of satisfied equations.

Equations of the form $av + \mathbf{b} \cdot \mathbf{w} = c$ where $v \in V^i$ is unbalanced and satisfies Eq. 4.35, and $\mathbf{w} \cap V^i = \emptyset$. If we combine Eq. 4.35 and Eq. 4.59 we obtain the inequality

$$\frac{S(U^i, \pi, v \leftarrow j)}{\binom{t}{k-1}} \geq \frac{S(U^i, \pi, v \leftarrow j')}{\binom{t}{k-1}} + \varepsilon_3 - 2\varepsilon_2. \quad (4.63)$$

By the construction of the algorithm, the value chosen for v will thus be the correct value, provided that $\varepsilon_3 > 2\varepsilon_2$. It follows that the number of satisfied equations of this type cannot decrease.

Equations of the form $av + \mathbf{b} \cdot \mathbf{w} = c$ where $v \in V^i$ is *not* unbalanced, but still satisfies Eq. 4.35, and $\mathbf{w} \cap V^i = \emptyset$. In this case, the algorithm may select the wrong assignment to v . However, that cannot decrease the optimum value 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 Eq. 4.35, this implies that

$$\left| \frac{S(V \setminus V^i, H, v \leftarrow j')}{\binom{n-|V^i|}{k-1}} - \frac{S(V \setminus V^i, H, v \leftarrow j)}{\binom{n-|V^i|}{k-1}} \right| \leq 2\varepsilon_2. \quad (4.64)$$

Since there are at most $|V^i|$ different v that are not unbalanced, we can bound the decrease in the number of satisfied equations by

$$|V^i| (S(V \setminus V^i, H, v \leftarrow j') - S(V \setminus V^i, H, v \leftarrow j)) \leq \frac{2\varepsilon_2 n^k}{(k-1)!\ell}. \quad (4.65)$$

Equations of the form $av + \mathbf{b} \cdot \mathbf{w} = c$ where $v \in V^i$ does not satisfy Eq. 4.35, and $\mathbf{w} \cap V^i = \emptyset$. 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 $\varepsilon_1 p^k |V^i| n^{k-1}/k!$. Only

$\varepsilon_1 p |V^i| n^{k-1} / k!$ of these can be satisfied at the same time, and hence a bound on the decrease is

$$\frac{\varepsilon_1 p |V^i| n^{k-1}}{k!} = \frac{\varepsilon_1 p n^k}{k! \ell}. \quad (4.66)$$

Summing up, the total decrease is

$$p n^k / k! \ell^2 + 2 \varepsilon_2 n^k / (k-1)! \ell + \varepsilon_1 p n^k / k! \ell. \quad (4.67)$$

By choosing

$$\ell = 2p^2 / k! \varepsilon, \quad (4.68)$$

$$\varepsilon_1 = \varepsilon k! / 4, \quad (4.69)$$

$$\varepsilon_2 = \varepsilon (k-1)! / 8, \quad (4.70)$$

$$\varepsilon_3 = \varepsilon (k-1)! / 3, \quad (4.71)$$

the total decrease becomes at most $\varepsilon n^k / p \ell$. \square

Corollary 4.22. *If we construct from an assignment $\pi = H|_U$, a new assignment Π_π as in step 3 of Algorithm 4.6, this new assignment has the property that $\mu(\Pi_\pi) \geq \mu(H) - \varepsilon/p$.*

Proof. Apply Lemma 4.12 ℓ times, one for each $i \in \{1, \dots, \ell\}$. \square

The main theorem follows by the same argument as Theorem 4.14

Theorem 4.23. *For instances of Max Ek-Lin mod p where the number of equations is $\Theta(n^k)$, Algorithm 4.6 is a randomized polynomial time approximation scheme.*

Proof. 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 4.18 and 4.22 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 $\varepsilon n^k / p$ for the equation problem. Since the optimum of an Max E2-Lin mod p instance with cn^k equations is at least cn^k / p , this gives a relative error which is at most ε/c . \square

4.4 Conclusions

We have shown how to construct a randomized polynomial time approximation scheme for dense instances of Max Ek-Lin 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.

It would be interesting to generalize the Max k -Function Sat problem to functions mod p , and see if dense instances of that problem admit a polynomial time approximation scheme. This should require only minor modifications of the algorithm and its analysis.

Bibliography

- [1] Paola Alimonti. Non-oblivious local search for graph and hypergraph coloring problems. In *Proceedings of 21st International Workshop on Graph-Theoretic Concepts in Computer Science*, volume 1017 of *Lecture Notes in Comput. Sci.*, pages 167–180. Springer-Verlag, Berlin, 1995.
- [2] Gunnar Andersson and Lars Engebretsen. An RPTAS for Max Ek Lin- p . Unpublished note, August 1997.
- [3] Gunnar Andersson and Lars Engebretsen. Better approximation algorithms for Set Splitting and Not-All-Equal Sat. *Information Processing Letters*, 65(6):305–311, April 1998.
- [4] Gunnar Andersson, Lars Engebretsen, and Johan Håstad. A new way to use semidefinite programming with applications to linear equations mod p . Manuscript, 1998.
- [5] Sanjeev Arora. Polynomial time approximation scheme for Euclidean TSP and other geometric problems. In *Proceedings of 37th Annual IEEE Symposium on Foundations of Computer Science*, pages 2–11. IEEE Computer Society, Los Alamitos, 1996.
- [6] Sanjeev Arora, David 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.
- [7] Stephen A. Cook. The complexity of theorem proving procedures. In *Proceedings of the Third Annual ACM Symposium on Theory of Comp*, pages 151–158. ACM, New York, 1971.
- [8] Pierluigi Crescenzi and Viggo Kann. A compendium of NP optimization problems. Technical Report SI/RR-95/02, Dipartimento di Scienze dell’Informazione, Università di Roma “La Sapienza”, 1995. Available in electronic form from [URL:http://www.nada.kth.se/theory/problemlist.html](http://www.nada.kth.se/theory/problemlist.html).

- [9] Pierluigi Crescenzi and Luca Trevisan. Max NP-completeness made easy. Technical Report TR97-039, Electronic Colloquium on Computational Complexity, September 1997.
- [10] 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 3rd Israel Symposium on Theory of Computing and Systems*, pages 182–189, 1995.
- [11] Wenceslas Fernandez de la Vega. MAX-CUT has a randomized approximation scheme in dense graphs. *Random Structures and Algorithms*, 9:93–97, 1996.
- [12] Alan Frieze and Mark Jerrum. Improved approximation algorithms for MAX k -CUT and MAX BISECTION. *Algoritmica*, 18:67–81, 1997.
- [13] Alan Frieze and Ravi Kannan. Quick approximation to matrices and applications. Manuscript, July 1997.
- [14] Michael R. Garey and David S. Johnson. *Computers and Intractability: a guide to the theory of NP-completeness*. W. H. Freeman and Company, San Francisco, 1979.
- [15] 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.
- [16] Michel X. Goemans and David P. Williamson. New $\frac{3}{4}$ -approximation algorithms for the maximum satisfiability problem. *SIAM Journal of Discrete Mathematics.*, 7(4):656–666, 1994.
- [17] Michel X. Goemans and David P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the Association of Computing Machinery*, 42:1115–1145, 1995.
- [18] Oded Goldreich, Shafi Goldwasser, and Dana Ron. Property testing and its connection to learning and approximation. In *Proceedings of 37th Annual IEEE Symposium on Foundations of Computer Science*, pages 339–348. IEEE Computer Society, Los Alamitos, 1996.
- [19] 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.
- [20] David S. Johnson. Approximation algorithms for combinatorial problems. *Journal of Computer and System Sciences*, 9:256–278, 1974.

- [21] 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.
- [22] Leonid A. Levin. Universal'nye zadachi perebora. *Problemy peredači informacii*, 9(3):115–116, 1973. In Russian.
- [23] László Lovász. Coverings and colorings of hypergraphs. In *Proceedings 4th Southeastern Conference on Combinatorics, Graph Theory, and Computing*, pages 3–12. Utilitas Mathematica Publishing, Winnipeg, 1973.
- [24] Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms*. Cambridge University Press, Cambridge, 1995.
- [25] Christos H. Papadimitriou and Mihalis Yannakakis. Optimization, approximation, and complexity classes. *Journal of Computer and System Sciences*, 43:425–440, 1991.
- [26] Erez Petrank. The hardness of approximation: Gap location. *Computational Complexity*, 4:133–157, 1994.
- [27] Gregory B. Sorkin. Personal communication.
- [28] Luca Trevisan, Gregory B. Sorkin, Madhu Sudan, and David P. Williamson. Gadgets, approximation, and linear programming. In *Proceedings of 37th Annual IEEE Symposium on Foundations of Computer Science*, pages 617–626. IEEE Computer Society, Los Alamitos, 1996.