KTH Numerical Analysis
and Computer Science

Beating a Random Assignment

Approximating Constraint Satisfaction Problems

GUSTAV HAST

Doctoral Thesis
Stockholm, Sweden 2005

TRITA-NA-0513
ISSN 0348-2952
ISRN KTH/NA/R--05/13--SE
ISBN 91-7178-051-3

KTH Numerisk analys och datalogi
SE-100 44 Stockholm
SWEDEN

Akademisk avhandling som med tillstånd av Kungl Tekniska högskolan framlägges till offentlig granskning för avläggande av teknologie doktorsexamen tisdagen den 14 juni 2005 klockan 14.15 i Sal E3, Kungl Tekniska högskolan, Osquars backe 14, Stockholm.

Tryck: Universitetsservice US AB

Abstract

An instance of a Boolean constraint satisfaction problem, CSP, consists of a set of constraints acting over a set of Boolean variables. The objective is to find an assignment to the variables that satisfies all the constraints. In the maximization version, Max CSP, each constraint has a weight and the objective is to find an assignment such that the weight of satisfied constraints is maximized. By specifying which types of constraints that are allowed we create subproblems to Max CSP. For example, an instance of Max kCSP only contains constraints that act over at most k different variables. Another problem is Max CSP(P), where P is a predicate, i.e., a Boolean function. In such an instance P is used to determine if a constraint is satisfied or not.

Both Max kCSP and Max CSP(P) are NP-hard to solve optimally for $k \geq 2$ and predicates P that depend on at least two input values. Therefore, we consider efficient approximation algorithms for these two problems. A trivial algorithm is to assign all variables a random value. Somewhat surprisingly, Håstad showed that using this random assignment approach is essentially optimal for approximating Max CSP(P), for some predicates P. We call such predicates *approximation resistant*. Goemans and Williamson introduced an approximation method that relaxes problems into semidefinite programs. Using this method they show that for predicates P of arity two, it is possible to outperform a random assignment in approximating Max CSP(P). By extending this technique Zwick characterized all predicates of arity three as either approximation resistant or not.

In this thesis we consider predicates of arity larger than three. We extend the work of Håstad and the work of Samorodnitsky and Trevisan in order to show predicates to be approximation resistant. We also use semidefinite relaxation algorithms in order to show that predicates are not approximation resistant. In particular we show that predicates with few non-accepting inputs are approximation resistant and that predicates with few accepting inputs are not approximation resistant. We study predicates of arity four more closely and characterize 354 out of 400 predicate types.

Max kCSP is 2^{-k}-approximated by a random assignment and previously no algorithms were known to outperform such an algorithm with more than a small constant factor. In this thesis a probabilistic $\Omega(2^{-k+\log k-\log\log k})$-approximation for Max kCSP is presented.

Sammanfattning

En instans av ett villkorssatisfieringsproblem, CSP, består av en mängd villkor som agerar över en mängd binära variabler. Uppgiften består av att hitta en tilldelning till variablerna så att alla villkor uppfylls. I maximeringsversionen, MAX CSP, har alla villkor en vikt och uppgiften består i att hitta en tilldelning så att vikten av uppfyllda villkor maximeras. Genom att specificera vilka typer av villkor som är tillåtna skapar man delproblem till MAX CSP. Ett exempel är MAX kCSP där endast villkor som agerar över som mest k variabler är tillåtna. Ett annat problem är MAX CSP(P), där P är ett predikat. I en sådan instans så används P för att avgöra om ett villkor är uppfyllt eller ej.

Problemen MAX kCSP och MAX CSP(P) är NP-svåra att lösa optimalt om $k \geq 2$ och predikatet P beror på åtminstone två indatavärden. De är därför meningsfullt att betrakta approximationsalgoritmer för dessa problem. En trivial algoritm är att tilldela varje variabel ett slumpmässigt valt värde. Håstad visade att en sådan slumptilldelning är i praktiken det bästa man kan göra för att approximera MAX CSP(P), för vissa predikat P. Sådana predikat kallar vi för approximationsresistenta. Goemans och Williamson introducerade en metod som bygger på semidefinit relaxering. Genom att använda denna kunde de visa att för alla predikat P som tar två indata kan man skapa en approximationsalgoritm för MAX CSP(P) som är bättre än en slumptilldelning. Genom att utvidga denna metod kunde Zwick karaktärisera varje predikat, som maximalt tar tre indata, som antingen approximationsresistent eller ej approximationsresistent.

I denna avhandling studerar vi predikat som tar fler än tre indata. Genom att utvidga resultat av Håstad samt resultat av Samorodnitsky och Trevisan visar vi att ett antal predikat är approximationsresistenta. Vi använder också algoritmer som bygger på semidefinit relaxering för att visa att predikat inte är approximationsresistenta. Vi visar att predikat med få icke-accepterande indata är approximationsresistenta och att predikat med få accepterande indata inte är approximationsresistenta. Vi undersöker speciellt predikat med fyra indata där vi karaktäriserar 354 av 400 predikattyper.

För MAX CSP(P) försöker vi främst skapa algoritmer som är bättre än en slumptilldelning. Fokus ligger inte på att göra dem så bra som möjligt. För MAX kCSP försöker vi däremot slå en slumptilldelning med en så stor faktor som möjligt. En slumptilldelning 2^{-k}-approximerar MAX kCSP. Vi ger en probabilistisk $\Omega(2^{-k+\log k-\log\log k})$-approximationsalgoritm för MAX kCSP som är den första som slår en slumptilldelning med en faktor som ökar med större k.

Acknowledgments

First of all, I would like to express my deepest gratitude and appreciation to my supervisor Johan Håstad. I have been extremely lucky in having Johan as a teacher and guide. A visit at Johan's office always results in new insights. The typical input of such a visit is some hard to express ideas and the output is a much clearer view of things. Despite a busy schedule, it is hard to remember a single time Johan has not been able to make time for helping me.

During the first year of my studies, Johan was on leave. During this time, Mikael Goldmann was my assistant supervisor. I thank Mikael for this and also for discussing research related problems. Apart from Johan and Mikael, I have had enlightening discussions relating to topics in this thesis with Lars Engebretsen, Jonas Holmerin and Rafael Pass. I thank them for this.

During my time as a PhD student at the theoretical computer science group at Nada I have been lucky to have had great room mates: Rafael Pass, Anna Redz, Magnus Rosell and Mårten Trolin. A special thanks to Rafael for sharing his enthusiasm and positive attitude towards science.

Per Austrin read a large part of this thesis and suggested several improvements. Thanks Per!

Finally, I thank Åsa.

Contents

Chapter 1

Introduction

A central question in computer science is which problems we can hope to solve by using a computer. An example of a real life problem is the following: "If Adam has three \$100 bills, four \$20 bills, five \$10 bills and eight \$1 dollar bills, can he use these to pay exactly \$125?". The answer is obviously "yes", but a program that simply outputs "yes" without looking at the problem cannot really be considered to have solved it. Instead we consider the above to be an instance of a problem. In order for a computer program, or an algorithm, to solve a problem we require it to be able to compute the correct result for all possible instances of the problem. The above question can for example be considered as an instance of the subset sum problem: "Given an integer S and n objects whose values are positive integers a_1, a_2, … a_n, is it possible to produce a subset of the objects which has total value S?".

The subset sum problem can be solved by an algorithm that calculates the value of every possible subset of the n elements. It outputs "yes" if it finds a subset of value S and otherwise outputs "no". The subset sum problem is said to be solvable as there exists an algorithm that outputs the correct answer for every instance. Perhaps rather counterintuitively, there do exist problems that cannot be solved by an algorithm. The most famous is the halting problem: "Given an algorithm, does it ever halt?". This was shown to be unsolvable by Turing [40] in the 1930s. But if a problem is solvable, does it really mean that it is reasonable to expect that we can find the answer using a computer? Adam has a total of 20 objects, thus there are $2^{20} \approx 1000000$ different subsets that we have to consider. Using a modern computer, this should not take much time. However, if an instance contains 80 objects, then calculating the value of every possible subset takes millions of years for a modern computer. We see that the distinction between solvable and unsolvable problems is not a practical one. In this thesis we instead consider the distinction between efficiently solvable problems and problems that cannot be solved efficiently.

1.1 P, NP and NP-Complete Problems

Even though there exist plenty of natural problems that are considered hard to solve, i.e., take long time, there do not always exist results that directly reflect this. It has over time been proven that for many seemingly hard problems it is difficult to produce results on the following form: “An algorithm that solves problem X has to perform at least $f(n)$ computations”, where n is the size of the instance and f is some fast increasing function. A way to cope with this situation is to relate the hardness of one problem with the hardness of other problems. This is what the theory of NP-completeness enables us to do.

The class P contains all decision problems that can be solved in polynomial time. A decision problem asks questions that can only be answered with “yes” or “no”. A problem is said to be solved in polynomial time if there exists an algorithm that, on every possible instance, produces the correct answer and the running time is upper bounded by a polynomial $p(n)$, where n is the size of the instance. In theoretical computer science, an efficient algorithm is normally meant to be an algorithm with a running time that is polynomially bounded. The class NP contains all decision problems where a “yes”-instance can be verified in polynomial time. By this we mean that there exists a proof that the “yes”-instance is indeed a “yes”-instance, and that this proof can be verified in polynomial time. For example, subset sum is in NP, where the proof simply specifies a subset of objects with the correct value.

Clearly $\mathrm{P} \subseteq \mathrm{NP}$, because if we can solve a problem, then we can also verify “yes”-instances of the problem by simply solving the instance. We do not even need a proof in such cases. The question whether $\mathrm{P} = \mathrm{NP}$ is an open one, but it is widely believed that there exists problems in NP that cannot be solved in polynomial time. In the beginning of the 1970s, Cook [5] introduced the notion of NP-complete problems, which are the hardest problems in NP. If any NP-complete problem can be solved in polynomial time, then all problems in NP can be solved in polynomial time, implying that $\mathrm{P} = \mathrm{NP}$. Since then, many problems, among these subset sum, have been shown to be hard using the theory of NP-completeness. In 1979 Garey and Johnson [12] published a book containing a long list of NP-complete problems.

1.2 Approximation Algorithms

The main problems we consider in this thesis are so-called constraint satisfaction problems, CSPs. For a CSP, the objective is to satisfy a collection of constraints by finding a good assignment. A natural decision problem is to ask if there exists an assignment that satisfies all constraints. However, we instead consider the maximization version of the problem. In Max CSP each constraint has a weight and the objective is to maximize the weight of satisfied constraints. By restricting what types of constraints are allowed, different types of problems can be defined. Two well-known examples are Max CUT and Max E3SAT.

A constraint in a Max CUT instance consists of two Boolean variables, x_i and x_j, and it is satisfied if $x_i \neq x_j$. Constraints of Max E3SAT are disjunctions over exactly three literals, where a literal is a variable or a negated variable. Both these problems are NP-hard, meaning that if we can solve these problems efficiently, then we can also solve NP-complete problems efficiently. Thus, we need to settle for something less. We could, for example, relax the requirement that the algorithm should work for all instances, and try to design an efficient algorithm that instead works for most instances. However, we take another route here and relax the requirement to solve the problem optimally. An algorithm is said to be an α-approximation of a maximization problem if for every instance the expected value of the produced solution is at least αw_{opt}, where w_{opt} is the value of an optimal solution. This line of research was initiated by a paper in 1973 by David S. Johnson [26].

A naive approximation algorithm for Max CUT and Max E3SAT, as well as all Max CSPs, is simply assigning all variables a random value, without looking at the constraints. Let us analyze the expected performance of this random assignment algorithm. There are four different possible assignments of the variables x_i and x_j, and two of these satisfy the Max CUT constraint, $x_i \neq x_j$. The variables are assigned uniformly random values and thus each assignment is equally probable. The probability that a constraint is satisfied is thus $1/2$. If w_{tot} is the total weight of all constraints in the instance, then the expected value of a random assignment is $w_{\text{tot}}/2$, where the value of an assignment is the sum of the weights of satisfied constraints. There are eight possible assignments to three Boolean variables. A disjunction over these three variables is satisfied by all but one of these assignments. A Max E3SAT constraint is thus satisfied by a random assignment with probability $7/8$ and the expected value of such a solution is $7w_{\text{tot}}/8$. We know that $w_{\text{opt}} \leq w_{\text{tot}}$, thus picking a random assignment is a $1/2$-approximation of Max CUT and a $7/8$-approximation of Max E3SAT.

For a long time, no efficient algorithms that significantly outperformed the random assignment algorithm were known to exist for Max CUT, Max E3SAT and many other similar problems. Johnson considered Max E3SAT in his original paper on approximation algorithms and constructed a deterministic $7/8$-approximation. In 1976, Sahni and Gonzales [34] produced a deterministic $1/2$-approximation of Max CUT. A number of algorithms ameliorating low order terms of the approximation ratio for Max CUT was then produced [16, 25, 32, 41]. However, no algorithm was a $1/2 + \varepsilon$-approximation for any constant $\varepsilon > 0$. In 1994, a major breakthrough was made when Goemans and Williamson [14] constructed an efficient 0.878-approximation of Max CUT. The algorithm relaxes the Max CUT instance into a semidefinite program which is solved using methods from mathematical programming. Afterwards, the relaxed solution is rounded into a Max CUT solution. The work of Goemans and Williamson inspired many researchers to construct approximation algorithms for other maximization problems based on the same method. A far from complete list includes [10, 17, 27, 28, 31, 42, 44].

Some years before the paper of Goemans and Williamson another major breakthrough in approximation theory was taking place which ultimately would demonstrate a major difference between the approximability of Max CUT and Max E3SAT: While Max CUT can efficiently be approximated better than picking a random assignment, it is NP-hard to $7/8 + \varepsilon$-approximate Max E3SAT. In other words, there is no efficient algorithm that significantly outperforms an algorithm that just picks a random assignment, unless P = NP. The basis for this result, and many other hardness of approximability results, is the PCP-theorem of Arora et al. [2] that translates an arbitrary instance x of an NP problem L into a Max E3SAT instance $\phi(x)$, such that if x is a "yes"-instance of L, then all constraints in $\phi(x)$ can be satisfied and otherwise less than a fraction $c < 1$ of them can be satisfied simultaneously. By choosing an NP-complete problem L, we see that the PCP-theorem implies that Max E3SAT is NP-hard to c-approximate. This is because we can decide if x is a "yes"-instance of L by c-approximating $\phi(x)$. By using redundant encodings of $\phi(x)$, the gap between c and 1 can be amplified into showing that it is NP-hard to $7/8+\varepsilon$-approximate Max E3SAT [22]. Several other combinatorial optimization problems have been shown hard to approximate using the gap from the PCP theorem [3, 8, 9, 21].

Another way of showing hardness of approximation results for Max CSPs is through the use of gadgets. A gadget transforms a constraint of one kind into a set of constraints of another kind, thereby relating the approximability of the two problems. Gadgets have been used for a long time but was explicitly defined by Bellare et al. [3]. Trevisan et al. [39] then showed how to use linear programming in order to find optimal gadgets. Using this technique it was shown that it is NP-hard to $16/17 + \varepsilon$-approximate Max CUT, for any $\varepsilon > 0$ [22, 39]. An exciting recent result shows that the Goemans-Williamson algorithm for Max CUT is in fact optimal, assuming the validity of two conjectures [30], one of which has later been established.

1.3 Approximation Resistance

In the late 1970s, Schaefer [36] made a complete characterization of which types of CSPs that admitted polynomial time algorithms for deciding if an instance is satisfiable or not. This fundamental result was later extended by Creignou [6] and Khanna et al. [29] for Max CSPs and they showed that almost all Max CSPs are NP-hard to solve optimally, except for some very restrictive types. They also showed that a Max CSP either is solvable or there exists a constant $\epsilon > 0$ such that it is NP-hard to $1 - \epsilon$-approximate the problem.

In one sense this solved the problem of characterizing the approximability of Max CSPs: Some few problems are solvable and the rest are APX-complete. However, both Max CUT and Max E3SAT are APX-complete but, as described above, they exhibit very different properties regarding approximability. For Max CUT there exist good approximation algorithms whereas for Max E3SAT we essentially cannot do better than picking a random assignment.

A subproblem of Max CSP is Max CSP(P), where P is a predicate mapping k Boolean values onto $\{0,1\}$. An instance of Max CSP(P) contains a set of weighted constraints. Each constraint consists of k literals, where a literal is a variable that may or may not be negated. A constraint is satisfied by an assignment if P maps its literals onto 1. The objective is to maximize the weight of satisfied constraints. The result of Creignou [6] and Khanna et al. [29] shows that Max CSP(P) is NP-hard for all predicates that depend on at least two variables. The distinction we focus on is whether Max CSP(P) can be approximated better than using a random assignment. If it is NP-hard to substantially outperform a random assignment on Max CSP(P), then P is said to be *approximation resistant*. In this thesis we characterize many predicates as either being approximation resistant or not.

There are many reasons for studying approximation resistance. First and foremost, approximation resistance is a fundamental property of a predicate which determines if anything non-trivial can be done in approximating its Max CSP in polynomial time. It is thus an important structural question. We believe that understanding this concept is a key to comprehending what it is that makes some problems so hard to approximate. Approximation resistant predicates also play a fundamental role in the design of efficient probabilistical proofs.

There are no predicates of arity two that are approximation resistant [14], even if the input variables are non-Boolean [23]. Håstad [22] showed that if P depends on three Boolean inputs and accepts all odd parity inputs or all even parity inputs, then P is approximation resistant. Zwick [42] completed the characterization for predicates of arity three by producing approximation algorithms for Max CSP(P), for all other predicates P, that outperform a random assignment.

A striking observation is that predicates with many accepting inputs seem more apt to be approximation resistant. Consider the two most extreme cases: predicate kAND that only accepts one input and kOR that accepts all but one input. Max CSP(kAND) corresponds to maximizing the number of satisfied conjunctions, Max kConjSAT. Trevisan [37] devised an algorithm for this problem that outperforms a random assignment with a factor of 2. Thus, kAND is never approximation resistant, regardless of its arity. In the second case, Håstad showed that kOR is approximation resistant as long as $k \geq 3$.

In order to produce efficient probabilistical checkable proofs, a number of predicates with large arity and few accepting inputs have been shown to be approximation resistant. Samorodnitsky and Trevisan [35] showed that there exist approximation resistant predicates of arity $2s + s^2$ with only 2^{2s} accepting inputs. Engebretsen and Holmerin [7] extended this line of work by showing approximation resistant predicates of arity $s + s(s-1)/2$ with only 2^s accepting inputs.

1.4 Max kCSP

Another Max CSP that we consider in this thesis is Max kCSP. This is a very general type of CSP, where the only requirement on the constraints is that they act over at most k Boolean variables. Trevisan observed that the hardest case of Max kCSP is if all constraints are conjunctions, Max kConjSAT. This is because given an arbitrary constraint acting over k variables, it can be expressed as a set of conjunctions such that if the original constraint is satisfied by an assignment, then exactly one of the conjunctions is satisfied by the same assignment. However, if the original constraint is not satisfied by an assignment, then no conjunctions are satisfied by that same assignment. Using a linear relaxation approach, Trevisan 2^{1-k}-approximated Max kConjSAT [37]. Using the above observation this implies that Max kCSP can be 2^{1-k}-approximated.

By combining good approximation algorithms for Max kConjSAT, where $k \leq 4$, with a method to shrink the size of large conjunctions, Hast [18] produced a $2^{1.54-k}$-approximation for Max kConjSAT and thereby also for Max kCSP. In this thesis we give the first algorithm that is shown to outperform a random assignment with an increasing factor for larger k. Furthermore, this factor is almost linear in k.

1.5 Organization and Contributions of this Thesis

In Chapter 2, we give some needed background, including notation, problem definitions, algorithms and techniques that we use in the thesis. In Chapter 3 we show how to prove that parity on three variables is approximation resistant. This is a result from [22], but we include it because many of our original results are based on the methods used here.

In Chapter 4, we introduce a general technique for approximating Max CSP instances. Using this technique, we are able to show that many predicates are not approximation resistant, e.g. all predicates on $2s$ variables with at most $2s + 1$ accepting inputs are shown to be not approximation resistant.

We show in Chapter 5 that all predicates implied by a Samorodnitsky-Trevisan predicate are approximation resistant. From this follows that predicates with few non-accepting inputs are approximation resistant.

In Chapter 6, we try to characterize predicates of arity four as either approximation resistant or not. This is primarily done by applying the methods in the previous chapters. We succeed in characterizing 354 out of 400 different predicate types.

Some observations on the structure of approximation resistant predicates are made in Chapter 7. In particular, we show that every predicate has arbitrarily "close" predicates that are approximation resistant. A natural conjecture would be to assume that if a predicate P is implied by an approximation resistant predicate, then P is also approximation resistant. This is true for all predicates of arity at most three [42] as well as for all predicates implied by a Samorodnitsky-Trevisan predicate. However, we combine earlier results and prove that this conjecture is false.

In Chapter 8, we describe an algorithm that $\Omega(2^{-k+\log k-\log\log k})$-approximates MAX kCSP. This constitutes a major progress compared with the previous best algorithm [18]. Unlike earlier chapters we are here interested in with how much we can beat a random assignment and not only that we can do it.

The material in Chapters 4, 5 and 6 is based on an unpublished paper [20]. The larger part of the work in Chapter 7 is previously unpublished. The approximation algorithm for MAX kCSP in Chapter 8 is described in [19]. Theorem 6.8 is from [18].

Chapter 2

Background

2.1 Basic Notation

A predicate P of arity k maps elements from $\{\pm 1\}^k$ onto $\{0, 1\}$. For notational convenience we let input bits have value -1, denoting true, and 1, denoting false. If P accepts an input y, then $P(y) = 1$, otherwise $P(y) = 0$. Thus, the set of accepting inputs to P is denoted by $P^{-1}(1)$.

Logical AND, OR and XOR between two variables x and y are denoted $x \wedge y$, $x \vee y$ and $x \oplus y$ respectively. Logical equality between x and y is denoted $x \equiv y$.

For an integer k, we let the predicate kOR and kAND be Boolean OR and AND over k variables respectively. Boolean XOR over k variables is expressed by kXOR. If $k\text{XOR}(x_1, \ldots x_k) = 1$ then $(x_1, \ldots x_k)$ is said to have odd parity, and otherwise even parity. A literal is a Boolean variable or a negated Boolean variable. The negation of a Boolean variable x is denoted $\overline{x}$.

For a set U of variables we let $\{\pm 1\}^U$ denote the set of all possible assignments to these variables. Suppose $U \subseteq W$ and $x \in \{\pm 1\}^W$, then the restriction of x to the variables occurring in U is denoted by $x|_U$. For a set of assignments $\alpha \subseteq \{\pm 1\}^W$, $\pi^U(\alpha)$ is a set consisting of elements $x \in \{\pm 1\}^U$ such that $x = y|_U$ for some $y \in \alpha$. Similarly, $x \in \pi_2^U(\alpha)$ if and only if α contains an odd number of elements y such that $x = y|_U$. We omit the superscript of π when the identity of U is evident from the context.

Let α and β be sets. The symmetric difference, $(\alpha \cup \beta) \setminus (\alpha \cap \beta)$, is denoted by $\alpha \Delta \beta$.

The probability of an event A is denoted $\Pr[A]$. For a random variable X, $\mathrm{E}[X]$ denotes the expected value of X. By $[m]$ we denote the set $\{1, 2, \ldots m\}$.

We use the following definition of the sign function sgn:

$$\text{sgn}(t) \quad = \quad \begin{cases} 1 & \text{if } t \geq 0 \\ 0 & \text{if } t < 0 \end{cases} \quad .$$

2.2 Basic Inequalities

Proposition 2.1 (The Cauchy-Schwartz inequality). *For any real numbers $(a_i)_{i=1}^n$ and $(b_i)_{i=1}^n$,*

$$\left|\sum_{i=1}^n a_i b_i\right|^2 \leq \left(\sum_{i=1}^n |a_i|^2\right)\left(\sum_{i=1}^n |b_i|^2\right) .$$

Proposition 2.2 (Jensen's inequality). *If f is a convex function and $\sum_{i=1}^n \lambda_i = 1$, where $\lambda_i \geq 0$, then*

$$f\left(\sum_{i=1}^n \lambda_i x_i\right) \leq \sum_{i=1}^n \lambda_i f(x_i) .$$

2.3 Problem Definitions

The main problem we consider in this thesis is Max CSP(P). In this problem we are asked to try to maximize the weight of satisfied constraints, where the predicate P is used to decide whether a constraint is satisfied or not.

Definition 2.3. Let $P : \{\pm 1\}^k \to \{0,1\}$ be a predicate. An instance of Max CSP(P) consists of m weighted constraints, each one acting over a k-tuple of literals, $(z_{i1}, \ldots z_{ik})$, taken from the set $\{x_1, \ldots x_n, \overline{x}_1, \ldots \overline{x}_n\}$. All variables in such a tuple are assumed to be distinct. A constraint is satisfied if and only if P accepts its k-tuple. A solution is an assignment to $\{x_1, \ldots x_n\}$. The value of a solution is $\sum_{i=1}^m w_i P(z_{i1}, \ldots z_{ik})$, where w_i is the (non-negative) weight of the i:th constraint. The objective is to maximize this value.

Remark 2.4. The definition above is somewhat different from Zwick's definition [42] in that it does not allow P to act over constants or multiple occurrences of the same variable. We have changed the definition in order to ensure that the expected value of a random assignment is a certain constant fraction of the total weight for all possible instances of Max CSP(P).

We define the weight of an input for a solution to a Max CSP(P) instance.

Definition 2.5. The weight of an input $y = (y_1, \ldots y_k)$ for an assignment a to a Max CSP(P) instance I is

$$\sum_{i:(z_{i1},\ldots z_{ik})=y} w_i ,$$

where each constraint tuple $(z_{i1}, \ldots z_{ik})$ is evaluated on a.

Definition 2.6. Two predicates of arity k, P and P', are of the same *type* if and only if there is a permutation π on $[k]$ and $a \in \{\pm 1\}^k$ such that $P(x_1, \ldots x_k) = P'(a_1 x_{\pi(1)}, \ldots a_k x_{\pi(k)})$ for all $x \in \{\pm 1\}^k$.

If P and P' are of the same type, then a MAX CSP(P) instance can be expressed as a MAX CSP(P') instance by permuting and applying a bitmask to the constraint tuples, so clearly they are equivalent problems.

In MAX kCSP we consider arbitrary constraints over at most k variables.

Definition 2.7. An instance of MAX kCSP consists of a set $\{C_1, \dots C_m\}$ of constraints with associated non-negative weights $\{w_1, \dots w_m\}$ and a set of Boolean variables $X = \{x_1, \dots x_n\}$. Each constraint C_i consists of a Boolean function f_i of arity $h \leq k$ and a size h tuple of Boolean variables $(x_{i_1}, \dots, x_{i_h})$ where $x_{i_j} \in X$. A solution is an assignment to X and the objective value of the solution is the sum of the weights of the satisfied constraints. A constraint $C_i = (f_i, (x_{i_1}, \dots, x_{i_h}))$ is satisfied if and only if $f_i(x_{i_1}, \dots, x_{i_h})$ is true.

We define some special cases of MAX kCSP.

Definition 2.8. MAX kSAT is a special type of MAX kCSP. Each constraint is a disjunction of at most k literals from $X \cup \overline{X}$, where $\overline{X} = \{\overline{x} : x \in X\}$. Furthermore, MAX E$k$SAT is a special type of MAX kSAT where every constraint acts over exactly k literals.

Definition 2.9. MAX kCONJSAT is a special type of MAX kCSP. Each constraint is a conjunction of at most k literals from $X \cup \overline{X}$, where $\overline{X} = \{\overline{x} : x \in X\}$.

Definition 2.10. MAX kALLEQUAL is a special type of MAX kCSP. Each constraint is a tuple of at most k literals from $X \cup \overline{X}$, where $\overline{X} = \{\overline{x} : x \in X\}$. A constraint is satisfied if and only if all literals have the same value.

We also need some definitions of formulas on *conjunctive normal form*, CNF.

Definition 2.11. A CNF*-formula* is a formula ϕ of n Boolean variables $\{x_i\}_{i=1}^n$ given by a conjunction of m clauses $\{C_j\}_{j=1}^m$. A clause contains a number of literals and it is true if at least one of the literals is true. The number of literals in a clause is the length of the clause.

Definition 2.12. A CNF-formula is *ρ-satisfiable* if some assignment satisfies at least ρm clauses.

Definition 2.13. A CNF-formula is an EkCNF*-formula* if each clause is of length exactly k.

Definition 2.14. A EkCNF-formula is an EkCNF*(l)-formula* if all variables occur in exactly l clauses.

2.4 Arithmetizing Predicates

A predicate can be seen as a sum of conjunctions, each conjunction corresponding to an accepting input to P. If $x, y \in \{\pm 1\}^k$ then $\sum_{S \subseteq [k]} \prod_{i \in S} x_i y_i$ equals 2^k if $x = y$

and otherwise the sum is zero. Thus, a conjunction $(x_1 \equiv \alpha_1) \wedge \ldots \wedge (x_k \equiv \alpha_k)$, where $\alpha \in \{\pm 1\}^k$, can be arithmetized as

$$\psi_\alpha(x_1, \ldots x_k) \;=\; 2^{-k} \sum_{S \subseteq [k]} \prod_{i \in S} \alpha_i x_i \;=\; \begin{cases} 1 & \text{if } \alpha = (x_1, \ldots x_k) \\ 0 & \text{otherwise} \end{cases} \quad . \tag{2.1}$$

A predicate can thus be expressed as a multilinear expression

$$P(x_1, \ldots x_k) \;=\; \sum_{\alpha \in P^{-1}(1)} \psi_\alpha(x_1, \ldots x_k) \; . \tag{2.2}$$

We let $P^{(i)}$ denote the sum of the i-degree terms of the multilinear expression P, and $P^{(\geq i)}$ denotes the sum of the terms of at least degree i in P. Thus,

$$P^{(i)}(x_1, \ldots x_k) \;=\; 2^{-k} \sum_{\alpha \in P^{-1}(1)} \sum_{S \subseteq [k], |S| = i} \prod_{i \in S} \alpha_i x_i \; .$$

2.5 Approximability

The sum of all weights in an instance I, $\sum_{i=1}^m w_i$, is denoted by $w_{\text{tot}}(I)$. We use $w_{\text{opt}}(I)$ to refer to the value of an optimal solution to the instance I. Sometimes we omit I, if the identity of the instance is evident from the context.

We use the following definition in order to determine the quality of an approximation algorithm.

Definition 2.15. An algorithm A *α-approximates* a maximization problem if for all instances I of the problem

$$w(A, I)/w_{\text{opt}}(I) \geq \alpha \; ,$$

where $w(A, I)$ is the value of the solution produced by A on input I and $w_{\text{opt}}(I)$ is the value of an optimal solution to I. Equivalently, A is said to have an *approximation ratio* of α. For probabilistic algorithms $w(A, I)$ is allowed to be an expected value over the random choices made by A.

Definition 2.16. An instance I of a Max CSP is *ρ-satisfiable* if and only if $\rho \leq w_{\text{opt}}(I)/w_{\text{tot}}(I)$. Furthermore, a solution to I is *ρ-satisfying* if it has value at least $\rho w_{\text{tot}}(I)$.

A predicate P is said to be *approximation resistant* if it is NP-hard to find a solution to a Max CSP(P) instance that is significantly better than the expected value of a random assignment. As a random assignment satisfies an arbitrary P-constraint with probability $2^{-k}|P^{-1}(1)|$, we have the following definition.

Definition 2.17. A predicate $P : \{\pm 1\}^k \to \{0, 1\}$ is *approximation resistant* if, for every constant $\epsilon > 0$, it is NP-hard to find a solution x to an instance I of Max CSP(P), such that the value of x is at least $(2^{-k}|P^{-1}(1)| + \epsilon) w_{\text{opt}}(I)$.

If a predicate is not approximation resistant, we say that it is non-trivially approximable.

Throughout this thesis we make the following assumption.

Assumption 2.18. A predicate $P : \{\pm 1\}^k \to \{0,1\}$ is not approximation resistant if, for some constant $\epsilon > 0$, there exists a polynomial time algorithm that can approximate Max CSP(P) within $2^{-k}|P^{-1}(1)| + \epsilon$.

We note that "approximate" in the above assumption refers to our definition that allows probabilistical algorithms and Definition 2.17 only requires it to be NP-hard to deterministically outperform a random assignment. However, if we can $2^{-k}|P^{-1}(1)| + \epsilon$-approximate Max CSP($P$), then by running the algorithm $2/\epsilon$ times we know that the best solution found has an approximation ratio of at least $2^{-k}|P^{-1}(1)| + \epsilon/2$ with probability at least $1/2$. Thus, if Assumption 2.18 is false, then we can solve an NP-hard problem in randomized polynomial time.

We also define predicates that are *hereditary approximation resistant.*

Definition 2.19. A predicate $P : \{\pm 1\}^k \to \{0,1\}$ is *hereditary approximation resistant* if all predicates P' implied by P, i.e., $(P(y) = 1) \Rightarrow (P'(y) = 1)$ for all inputs y, are approximation resistant.

We introduce the gain of an assignment in order to quantify how an assignment compare to the expected value of a random assignment.

Definition 2.20. The *gain* δ of an assignment a on an instance I is

$$\delta = \frac{\mathrm{val}(a, I) - w_{\mathrm{rand}}(I)}{w_{\mathrm{tot}}(I)} ,$$

where $\mathrm{val}(a, I)$ is the value of a on I and $w_{\mathrm{rand}}(I)$ is the expected value of a random assignment on I.

2.6 Probabilistically Checkable Proofs

We are concerned with the following question: How can a computational unbounded prover convince us of the validity of a statement $v \in L$? As mentioned in the introduction, if L is in the class NP, then we know that there exists a proof of the statement $v \in L$ that convinces us. In the case of subset sum, the proof specified which subsets to choose and it was easy to verify the validity of the proof.

Here we consider a somewhat different setting where the verifier is probabilistic and only looks at some bits of the proof. Such a proof is called a probabilistical checkable proof, or simply a PCP. For a probabilistic verifier there are two vital parameters, the completeness c and the soundness s. The completeness is a lower bound on the probability that the verifier accepts a valid proof and the soundness is an upper bound on the probability that the verifier accepts a proof of a false statement. Two other parameters associated with a verifier is the amount of random bits it uses and the number of bits in the proof it looks at.

We say that a language L belongs to $\mathrm{PCP}_{c,s}[r, q]$ if there exists a probabilistical polynomial time verifier V that only looks at q bits in the proof, uses at most r random bits and

1. for every $x \in L$, there exists a proof Π such that $\Pr[V \text{ accepts } (\Pi, x)] \geq c$,

2. for every $x \notin L$ and every proof Π, $\Pr[V \text{ accepts } (\Pi, x)] < s$.

In the above definition a verifier is allowed to first look at a bit in the proof and then, depending on the value of that bit, choose another bit to look at. We call such a verifier adaptive. A non-adaptive verifier has to choose which bits to look at before actually looking at a bit.

There exists a close connection between hardness of approximation and PCPs. This can be seen by regarding each bit in the proof as a variable and the problem of finding a proof that maximizes the accept probability of the verifier is then reduced to a Max CSP instance. The verifier inspects, for every possible outcome of its internal coin flips, q bits and decides whether to accept or reject. For each such outcome we introduce a constraint in a Max CSP instance, such that the constraint is satisfied if and only if the verifier accepts. Theorem 2.22 and its proof makes this connection between inapproximability and PCPs more explicit.

If a verifier in a PCP always uses a predicate P in order to decide whether to accept or reject, then the resulting proof optimization problem is a Max CSP(P) instance. We say that such verifiers have acceptance condition P.

Definition 2.21. A verifier with *acceptance condition* P, where P is a predicate mapping elements from $\{\pm 1\}^k$ onto $\{0, 1\}$, is a non-adaptive verifier that chooses k bits to look at from the proof, and applies P on the, possibly negated, bits from the proof. If P evaluates to 1, the verifier accepts and otherwise it rejects.

If we have an efficient PCP with acceptance condition P, then P is approximation resistance. By efficient we mean that the completeness can be made arbitrarily close to 1 and the soundness can be made arbitrarily close to the acceptance probability of a random proof.

Theorem 2.22 (Folklore). *Let $P : \{\pm 1\}^k \to \{0, 1\}$ be a predicate and L be an* NP*-complete language. If, for every constant $\epsilon > 0$, there exists a polynomial time* PCP *verifier for L, with acceptance condition P and that uses logarithmic randomness and has at least completeness $1 - \epsilon$ and at most soundness $2^{-k}|P^{-1}(1)| + \epsilon$, then P is approximation resistant.*

Proof. It is NP-hard to decide if an arbitrary element v belongs to L. We show that if we can find, given an instance I of Max CSP(P), a solution of value at least $(2^{-k}|P^{-1}(1)| + \gamma)w_{\mathrm{opt}}(I)$, for an arbitrary constant $\gamma > 0$, then we can also decide whether v belongs to L. This then implies that P is approximation resistant.

We set ϵ to a value such that

$$\frac{2^{-k}|P^{-1}(1)| + \epsilon}{1-\epsilon} \quad < \quad 2^{-k}|P^{-1}(1)| + \gamma \ .$$

The verifier uses a logarithmic number of random bits, thus there are only a polynomial number of possible outcomes. For each such outcome the verifier queries k bits from the proof and test these by applying P. We add the corresponding P-constraint to a Max CSP(P) instance I. We note that the probability that the verifier accepts a proof Π is exactly the fraction of satisfied constraints in I by the assignment defined by Π.

The soundness of the PCP implies that if $v \notin L$, then no assignment exists that satisfies a $2^{-k}|P^{-1}(1)| + \epsilon$ fraction of the constraints in I. On the other hand, if $v \in L$ then the completeness of the PCP implies that I is $1-\epsilon$-satisfiable. If $v \in L$, then by $2^{-k}|P^{-1}(1)| + \gamma$-approximating I we find a solution of value at least

$$\begin{aligned}(2^{-k}|P^{-1}(1)| + \gamma)w_{\text{opt}}(I) \quad &\geq \quad (2^{-k}|P^{-1}(1)| + \gamma)(1-\epsilon)w_{\text{tot}}(I) \\ &> \quad (2^{-k}|P^{-1}(1)| + \epsilon)w_{\text{tot}}(I) \ .\end{aligned}$$

As this value is larger than the upper bound on $w_{\text{opt}}(I)$ in the case $v \notin L$ we conclude that we can decide whether v belongs to L or not. □

2.7 Semidefinite Relaxation Algorithms

A successful approach in designing approximation algorithms for various problems has been to use relaxations. By relaxing the problem, we allow a wider range of solutions. Due to the extension of the solution space, the problem is then tractable to solve. Generally an optimal solution to the relaxed problem is not an admissible solution to the original problem. Therefore, a method in order to transform a relaxed solution into an admissible one is needed. A classical example of a relaxation is integer programming that is relaxed into linear programming by allowing variables to be real numbers instead of integers.

After Goemans and Williamson introduced their approximation algorithm for Max CUT, a long line of semidefinite relaxation algorithms have been proposed. Many results in this thesis relies on that there exists a good algorithm for approximating light instances of Max 2AllEqual. A light instance is an instance where the value of an optimal solution is close to the expected value of a random solution. For such instances it makes sense to approximate the gain instead. The algorithm of Charikar and Wirth allows us to do this. Another semidefinite relaxation algorithm that we use in this thesis is Zwick's algorithm for almost satisfiable Max 2SAT instances.

The Algorithm of Charikar and Wirth

We have the following quadratic programming problem, Max QP: Given a real valued matrix A, with null diagonal entries, maximize

$$\sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}x_ix_j, \text{ such that } x_i \in \{\pm 1\} \ . \tag{2.3}$$

We show that this is in fact a shifted case of Max 2AllEqual. Given a Max 2AllEqual instance I we can produce a Max QP instance by for every constraint $x_i = x_j$ with weight w_{ij} setting $a_{ij} := w_{ij}$ and for every constraint $\overline{x}_i = x_j$ with weight w_{ij} setting $a_{ij} := -w_{ij}$. All other elements in A are set to 0. Let the value of a solution x to the Max QP instance be $q(x)$ and the value of x to the Max 2AllEqual instance be $c(x)$. If a constraint involving x_i and x_j is satisfied, then $a_{ij}x_ix_j$ gives contribution w_{ij} to $q(x)$ and if it is not satisfied, then $a_{ij}x_ix_j = -w_{ij}$. Therefore we have

$$q(x) = c(x) - (w_{\text{tot}}(I) - c(x)) = 2c(x) - w_{\text{tot}}(I) = 2\delta w_{\text{tot}}(I) \ , \tag{2.4}$$

where δ is the gain of x on I. An algorithm that α-approximates Max QP also α-approximates the gain of Max 2AllEqual, since the gain is a constant multiple of $q(x)$.

Charikar and Wirth [4] created a semidefinite relaxation of Max QP. They used the standard technique, introduced by Goemans and Williamson [14], where each variable x_i is relaxed into an n-dimensional vector v_i of unit length. The product x_ix_j then becomes the inner product between v_i and v_j. The relaxation of (2.3) is then

$$\sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}v_i \cdot v_j, \text{ such that } v_i \cdot v_i = 1 \text{ for all } i \in [n] \text{ and } v_i \in \mathbb{R}^n \ . \tag{2.5}$$

It is a relaxation as an arbitrary solution $(x_i)_{i=1}^n$ can be transformed into vectors $(v_i)_{i=1}^n$, where $v_i = (x_i, 0, \ldots 0)$, such that the value of $(x_i)_{i=1}^n$ on (2.3) is the same as the value of $(v_i)_{i=1}^n$ on (2.5). Maximizing (2.5) is an example of semidefinite programming, which can be approximated efficiently within an arbitrary additive constant, see [1].

A good relaxation is only the first part of a successful approximation algorithm. By solving the semidefinite program we obtain a vector solution $(v_i)_{i=1}^n$ which essentially is optimal. There now has to be a method of obtaining a solution $(x_i)_{i=1}^n$ from this vector solution, such that if $(v_i)_{i=1}^n$ gives a large value to (2.5), then $(x_i)_{i=1}^n$ should give a large value to (2.3).

In order to create the $(x_i)_{i=1}^n$ solution, a random vector r on the hypersphere in $\mathbb{R}^n$ is chosen. The value of a variable x_i is then decided by a flip of a biased coin, where the bias is decided by the value of $r \cdot v_i$. Analyzing this rounding technique give us the following lemma.

Lemma 2.23 (Charikar and Wirth [4]). *If δ^* is the optimum gain of a* Max 2AllEqual *instance,* ApproxMaxQP *returns a solution whose expected gain is at least*

$$c_{\text{cw}} \left(\frac{\delta^*}{\log(1/\delta^*)} \right) \quad ,$$

for a constant $c_{\text{cw}} > 0$.

The proof of Lemma 2.23 shows that $c_{\text{cw}} = 1/64$ is a valid value to the constant. The algorithm *ApproxMaxQP* is a probabilistical polynomial time algorithm that acts as described above. We note that the original lemma instead considers Max CUT, however it works as well for the more general Max 2AllEqual problem. The switch does not effect the analysis in any harmful way.

This algorithm works well for Max 2AllEqual instances where the value of an optimal assignment is close to the expected value of a random assignment. We note that Zwick [44] and Feige and Langberg [10] also have produced algorithms for this problem which work in a very similar fashion.

Zwick's Max 2SAT Algorithm for Almost Satisfiable Instances

The Charikar-Wirth algorithm works well for instances where the value of an optimal solution is not far from the expected value of a random solutions. If an optimal solution instead satisfies almost all constraints for some instance, then we can use specialized semidefinite relaxation algorithms in order to obtain a good solution. This is true for Max 2AllEqual but also for Max 2SAT, as the following theorem shows.

Theorem 2.24 (Zwick [43]). *Given a $(1-\epsilon)$-satisfiable* Max 2SAT *instance a $(1-5\epsilon^{1/3})$-satisfying assignment can be found in probabilistical polynomial time.*

2.8 Gadgets

A gadget is a way to transform a constraint of one type into a set of constraints of another type. The notion of a gadget was defined by Bellare et al. [3] but the method has been used for a long time. A well known example appears in the reduction from 3SAT to Max 2SAT by Garey et al. [13].

In this work we consider gadgets from the parity check predicate 3XOR in order to prove hardness of approximation results. A gadget from 3XOR to a predicate $P : \{\pm 1\}^k \rightarrow \{0,1\}$ consists of a set of valid weighted constraints $\{(w_i, P(z_{i1}, \ldots z_{ik}))\}$ where the z_{ij}'s are literals of the variables x_1, x_2, x_3 or of the auxiliary Boolean variables y_1, …, y_l. Furthermore, it is an α-gadget if and only if the following holds:

1. If $3\text{XOR}(x_1, x_2, x_3) = 1$, then $\sum w_i P(z_{i1}, \ldots z_{ik}) = \alpha$ for some assignment to y_1, …, y_l.

2. If $3\text{XOR}(x_1, x_2, x_3) = 0$, then $\sum w_i P(z_{i1}, \ldots z_{ik}) \le \alpha - 1$ for all assignment to y_1, …, y_l.

For a more thorough look at gadgets, see [39].

Håstad showed that 3XOR is approximation resistant. A gadget from 3XOR to a predicate P therefore implies a hardness of approximation result for Max CSP(P). We have the following lemma.

Lemma 2.25 (Håstad [22])**.** *If there exists an α-gadget from* 3XOR *to a predicate P, then it is* NP*-hard, for any $\epsilon > 0$, to approximate* Max CSP(P) *within* $1 - 1/2\alpha + \epsilon$.

Chapter 3

3XOR is Approximation Resistant

In this chapter we show how to achieve a strong inapproximability result, namely that parity on three variables, 3XOR, is approximation resistant. This result is due to Håstad [22]. We describe how it is derived because our hardness of approximation results in Chapters 5 and 6 are very much based on this work. Both by using similar proof techniques and exploiting the fact that 3XOR is approximation resistant in gadget constructions, i.e., applying Lemma 2.25.

In order to show that 3XOR is approximation resistant, we construct an efficient PCP verifier for an arbitrary NP-complete language with acceptance condition 3XOR and apply Theorem 2.22. We give a summary how the PCP is created.

1. We are given an element v and a language $L \in \text{NP}$. The PCP theorem, Theorem 3.1, gives a transformation ϕ, such that $\phi(v)$ is a E3CNF-formula which is satisfiable if $v \in L$ and otherwise it is not c-satisfiable, for a constant $c < 1$.

2. We introduce a two-prover game between a verifier and two provers for verifying that a E3CNF-formula is satisfiable. The probability that an unsatisfiable E3CNF-formula is accepted by the verifier is called the soundness. In order to reduce the soundness of the two-prover game we let the verifier ask parallel questions.

3. We construct a PCP such that the proof Π should be encodings of answers in the parallel two-prover game. The verifier uses at most a logarithmic (in $|v|$) amount of random bits and decides to look at three bit positions in the proof, i, j and k, and accepts if $\text{3XOR}(\Pi_i, \Pi_j, \Pi_k)$. The number of possible 3-tuples the verifier can choose is polynomial in $|v|$. If Π is an encoding of a satisfying assignment to $\phi(v)$ then the verifier accepts with probability $1 - \epsilon$. We can set ϵ to an arbitrarily small positive constant.

4. If the accept probability in the PCP is at least $(1 + \delta)/2$, where δ is an arbitrarily small positive constant, then we can create a strategy for the provers in

the above parallel two-prover game with a success probability that is larger than the soundness of the game. Thus, in this case we know that $\phi(v)$ is satisfiable and $v \in L$.

We have that $2^{-3}|3\text{XOR}^{-1}(1)| = 1/2$, thus we can apply Theorem 2.22 and conclude that 3XOR is approximation resistant.

3.1 PCP Theorem

Most strong inapproximability results are based on the PCP theorem.

Theorem 3.1 (PCP Theorem, Arora et al. [2]). *For a constant $c < 1$, there is a polynomial transformation ϕ that given a language $L \in \text{NP}$ and an element v produces a* E3CNF*(5)-formula $\phi(v)$ such that:*

- *$\phi(v)$ is satisfiable if $v \in L$, and*
- *$\phi(v)$ is not c-satisfiable if $v \notin L$.*

The version of the PCP theorem we use, where $\phi(v)$ is a E3CNF(5)-formula, is due to Feige [8]. We note that the above is a PCP with perfect completeness and soundness c where the proof contains an assignment to the variables in $\phi(v)$ and the verifier picks a random clause and inspects its three variables. However, we would like a PCP with much lower soundness and a verifier that looks at three bits and decides to accept depending on the parity of these bits. In order to be able to produce such a PCP we first need to define a two-prover game.

3.2 Two-Prover Game

In a two-prover game a verifier V interacts with two different provers, P_1 and P_2. In Figure 3.1 a two-prover one round game is specified for deciding if a E3CNF-formula is satisfiable. If ϕ is satisfiable, then there exist two provers P_1 and P_2 such that $\Pr[V \text{ accepts}] = 1$. We say that the game has perfect completeness. If ϕ is not c-satisfiable, then for all possible provers P_1 and P_2 we have that $\Pr[V \text{ accepts}] < (2 + c)/3$. This is because the answers from P_2 defines an assignment x to the variables in ϕ. More than a $1 - c$ fraction of the clauses are not satisfied by this assignment. If V chooses one of these clauses, then P_1 has to give an answer that is inconsistent with x by changing the value of at least one variable. With probability at least $1/3$, P_1 changed the value of the variable that was sent to P_2, and then the verifier rejects. This happens with probability more than $(1 - c)/3$, thus it accepts with probability less than $(2 + c)/3$. We say that the game has soundness $s = (2 + c)/3$.

In order to make the probability that the verifier rejects not satisfiable formulas higher we need to lower the soundness. This could be achieved by repeating the game u times, thereby reducing the soundness to s^u. However, in order to make

Input: A E3CNF-formula, $\phi = C_1 \vee C_2 \dots C_m$.

1. V chooses $k \in [m]$ uniformly at random.
2. V chooses a variable x_j uniformly at random from the variables appearing in C_k.
3. V sends k to P_1 receiving an answer $y \in \{\pm 1\}^3$.
4. V sends j to P_2 receiving an answer $z \in \{\pm 1\}$.
5. V accepts if z is consistent with y and $y \neq (1, 1, 1)$.

Figure 3.1: The two-prover game

the connection with a PCP we would like to have only one round in the two-prover game. Therefore, we define a game where the verifier asks u parallel questions to the provers. The game is defined in Figure 3.2.

Input: A E3CNF-formula, $\phi = C_1 \vee C_2 \dots C_m$.

1. V chooses for $i = 1, \dots u$: $k_i \in [m]$ uniformly at random.
2. V chooses for $i = 1, \dots u$: a variable x_{j_i} uniformly at random from the variables appearing in C_{k_i}.
3. V sends $(k_i)_{i=1}^u$ to P_1 receiving answers $y_i \in \{\pm 1\}^3$ for $i = 1, \dots u$.
4. V sends $(j_i)_{i=1}^u$ to P_2 receiving answers $z_i \in \{\pm 1\}$ for $i = 1, \dots u$.
5. V accepts if $(z_i)_{i=1}^u$ are consistent with $(y_i)_{i=1}^u$ and all $y_i \neq (1, 1, 1)$.

Figure 3.2: The parallel two-prover game

The soundness of the parallel game is not s^u, however due to Raz [33] we know that the soundness decreases exponentially with u.

Theorem 3.2 (Raz [33]). *Let ϕ be a* E3CNF*-formula which is at most c-satisfiable. Then there exists a constant $d_c < 1$, given by the value of c, such that d_c^u is an upper bound for the probability that the verifier accepts in the u-parallel game defined in Figure 3.2 with input ϕ.*

3.3 The PCP Proof

By using the parallel two-prover game the soundness has been reduced at the cost of long answers. The verifier should not have to inspect the whole answers, rather it should only look at three bits. In order to make this possible, all answers to the possible questions in the two-prover game are encoded using the long code. The long code is a highly redundant code introduced by Bellare et al. [3]:

Definition 3.3. The *long code* of an assignment $x \in \{\pm1\}^t$ consists of the values of $f(x)$ for all functions $f : \{\pm1\}^t \to \{\pm1\}$.

The number of functions $f : \{\pm1\}^t \to \{\pm1\}$ is 2^{2^t}, thus this is the length of an assignment on t variables encoded with the long code. A standard written proof should contain long code encodings of assignments to all sets of variables that the verifier may ask for.

Definition 3.4. A *Standard Written Proof* with parameter u or $\mathrm{SWP}(u)$, contains for each set $V \subset [n]$ of size at most $3u$ a string of length $2^{2^{|V|}}$ which we interpret as a long code of an assignment to the variables $(x_i)_{i \in V}$.

The parameter u corresponds to the number of parallel question that are made in the two-prover game.

Definition 3.5. A $\mathrm{SWP}(u)$ is a *correct proof* for a formula ϕ of n variables if there is an assignment x which satisfies ϕ such that A_V is the long code of $x|_V$ for any V of size at most $3u$.

The proof in the PCP is thus a $\mathrm{SWP}(u)$. There is no guarantee that an actual proof that the verifier checks, really is a correct proof and consists of long code tables of satisfying assignments. However, there are two things that we can do to limit the power of a cheating prover. We can fold the supposedly long code tables over true and condition them.

Folding

A long code A_V has the property that $A_V(-f) = -A_V(f)$. Given an arbitrary table A we can fold it over true thereby producing A_{true}. This is done by for each pair of functions $(f, -f)$ we choose one and set $A_{\text{true}}(f) = A(f)$ and $A_{\text{true}}(-f) = -A(f)$ if f was chosen. If the tables in the $\mathrm{SWP}(u)$ really are long codes, then they are unaffected by the folding.

Conditioning

The other way to limit the power of a cheating prover is to condition the tables upon a function h. The answer from prover P_1 in the u-parallel two-prover game should contain a satisfying assignment to u clauses. We let h be the function such that $h(x)$ is true if and only if x satisfies all u clauses. Given an arbitrary table A we can condition upon h thereby producing A_h by setting for each function f, $A_h(f) = A(f \wedge h)$. If A is a long code encoding of a satisfying assignment, then A is uneffected by conditioning upon h.

Fourier Analysis of Tables

In order to analyze tables of supposedly long codes we use the discrete Fourier transform. The basis functions are $\chi_\alpha(f) = \prod_{x\in\alpha} f(x)$ where $\alpha \subseteq \{\pm1\}^t$ and the inner product of two tables A and B are given by $2^{-2^t}\sum A(f)B(f)$, where a function $f : \{\pm1\}^t \to \{\pm1\}$ acts as an index to a table. The Fourier coefficients are now given by

$$\hat{A}_\alpha = 2^{-2^t}\sum_f A(f)\chi_\alpha(f)$$

and we have

$$A(f) = \sum_\alpha \hat{A}_\alpha\chi_\alpha(f) \ .$$

We note that if A is a long code of an assignment x, then $\hat{A}_{\{x\}} = 1$ and $\hat{A}_\alpha = 0$ for all $\alpha \neq \{x\}$. We also use that if A is folded over true, then

$$\hat{A}_\emptyset = 2^{-2^t}\sum_f A(f) = 0 \ . \tag{3.1}$$

Parseval's identity gives us the sum of all Fourier coefficients squared.

$$\sum_\alpha \hat{A}_\alpha^2 = 2^{-2^t}\sum_f (A(f))^2 \tag{3.2}$$

We mostly consider functions A that map elements to $\{\pm1\}$ in which case the above sum is equal to 1. From the definition of χ_α we have that

$$\chi_\alpha(fg) = \chi_\alpha(f)\chi_\alpha(g)$$

and

$$\chi_\alpha(f)\chi_\beta(f) = \chi_{\alpha\Delta\beta}(f) \ ,$$

where Δ is the symmetric difference. If f only depends on variables in $U \subset W$ and $\beta \subseteq \{\pm1\}^W$, then we have that

$$\chi_\beta(f) = \prod_{x\in\beta} f(x) = \prod_{x\in\pi_2^U(\beta)} f(x) \ .$$

We also have the following lemma.

Lemma 3.6. *For any non-empty $\alpha \subseteq \{\pm1\}^t$*

$$\mathrm{E}_f\left[\chi_\alpha(f)\right] = 0 \ ,$$

where the expectation is taken over randomly and uniformly chosen Boolean functions $f : \{\pm1\}^t \to \{\pm1\}$.

Proof. Let $x_0 \in \alpha$. For every function $f : \{\pm 1\}^t \to \{\pm 1\}$ such that $f(x_0) = 1$ there exists a single function f' such that $f'(x_0) = -1$, but $f(x) = f'(x)$ for all $x \neq x_0$. For such a pair of functions we have that

$$\chi_\alpha(f) + \chi_\alpha(f') = (f(x_0) + f'(x_0)) \prod_{x \in \alpha \setminus \{x_0\}} f(x) = 0 \ .$$

It is easy to see that the space of possible functions can be covered by non-overlapping pairs of functions as specified above. The lemma then follows. □

3.4 The PCP Verifier

In Figure 3.3 the actions of the verifier is described. We see that the accept criterion for the verifier can be expressed as a 3XOR of bits in the proof. This is important because each possible question the verifier may ask corresponds to a 3XOR constraint in the instance I we want to produce. First we show the completeness of

Input: A SWP(u).

1. The verifier chooses a set U of u variables and a random boolean function f on U. Let A be the portion of the proof corresponding to U. A is folded over true.
2. For each variable in U choose a random clause containing it. Let h be the conjunction of the chosen clauses and let W be the set of variables appearing in the chosen clauses. Choose g_1 to be a random boolean function on W. Let B be the portion of the proof corresponding to W. B is folded over true and conditioned upon h.
3. Choose a function μ on W which, independently at each point takes the value 1 with probability $1 - \epsilon$ and -1 with probability ϵ. Set $g_2 = f g_1 \mu$, i.e., define g_2 by for each $y \in \{\pm 1\}^W$, $g_2(y) = f(y|_U) g_1(y) \mu(y)$.
4. The verifier accepts if and only if

$$A(f)B(g_1)B(g_2) \quad = \quad 1 \ .$$

Figure 3.3: The PCP verifier

the PCP.

Lemma 3.7. *The completeness of the* PCP *is at least* $1 - \epsilon$.

Proof. Let the input be a correct proof produced from a satisfying assignment x. Then $A(f) = f(x|_U)$, $B(g_1) = g_1(x|_W)$ and $B(g_2) = f(x|_U) g_1(x|_W) \mu(x|_W)$. Thus, if $\mu(x|_W) = 1$ then $A(f)B(g_1)B(g_2) = 1$. This happens with probability $1 - \epsilon$. □

Soundness of the Verifier

More care is needed to establish the soundness of the PCP. In this section we show the following lemma.

Lemma 3.8. *Assume*

$$\mathrm{E}\left[A(f)B(g_1)B(g_2)\right] \;=\; \delta\ , \tag{3.3}$$

where the expectation is taken over all coin tosses of the PCP *verifier. Then there is a strategy for the two provers in the two-prover game that convinces the verifier with probability at least* $4\epsilon\delta^2$.

Proof. In order to prove the lemma we design randomized strategies for P_1 and P_2 that are based on the value of the Fourier coefficients of A and B. We start by applying the Fourier inversion formula on A and B in (3.3).

$$\begin{aligned}&\mathrm{E}_{f,g_1,\mu}\left[A(f)B(g_1)B(g_2)\right]\\ &= \mathrm{E}_{f,g_1,\mu}\left[\sum_{\alpha,\beta_1,\beta_2} \hat{A}_\alpha \hat{B}_{\beta_1}\hat{B}_{\beta_2}\chi_\alpha(f)\chi_{\beta_1}(g_1)\chi_{\beta_2}(g_2)\right]\\ &= \sum_{\alpha,\beta_1,\beta_2} \hat{A}_\alpha \hat{B}_{\beta_1}\hat{B}_{\beta_2}\mathrm{E}_{f,g_1,\mu}\left[\chi_\alpha(f)\chi_{\beta_1}(g_1)\chi_{\beta_2}(fg_1\mu)\right]\end{aligned} \tag{3.4}$$

We analyze the expected value in the above expression.

$$\begin{aligned}&\mathrm{E}_{f,g_1,\mu}\left[\chi_\alpha(f)\chi_{\beta_1}(g_1)\chi_{\beta_2}(fg_1\mu)\right]\\ &= \mathrm{E}_{f,g_1,\mu}\left[\chi_\alpha(f)\chi_{\pi_2(\beta_2)}(f)\chi_{\beta_1}(g_1)\chi_{\beta_2}(g_1)\chi_{\beta_2}(\mu)\right]\\ &= \mathrm{E}_f\left[\chi_{\alpha\Delta\pi_2(\beta_2)}(f)\right]\mathrm{E}_{g_1}\left[\chi_{\beta_1\Delta\beta_2}(g_1)\right]\mathrm{E}_\mu\left[\chi_{\beta_2}(\mu)\right]\end{aligned}$$

By Lemma 3.6 we know that in order for $\mathrm{E}_f\left[\chi_{\alpha\Delta\pi_2(\beta_2)}(f)\right]$ and $\mathrm{E}_{g_1}\left[\chi_{\beta_1\Delta\beta_2}(g_1)\right]$ to be non-zero, then $\alpha = \pi_2(\beta_2)$ and $\beta_1 = \beta_2$. Finally we show that the expectation of the last factor decreases exponentially with the size of β_2:

$$\mathrm{E}_\mu\left[\chi_{\beta_2}(\mu)\right] \;=\; \mathrm{E}_\mu\left[\prod_{y\in\beta_2}\mu(y)\right] \;=\; \prod_{y\in\beta_2}\mathrm{E}_\mu\left[\mu(y)\right] \;=\; (1-2\epsilon)^{|\beta_2|}\ .$$

Summing up, the only non-zero terms in (3.4) are the ones where $\beta_1 = \beta_2$ and $\alpha = \pi_2(\beta_2)$ reducing it to a single sum.

$$\mathrm{E}_{f,g_1,\mu}\left[A(f)B(g_1)B(g_2)\right] \;=\; \sum_\beta \hat{A}_{\pi_2(\beta)}\hat{B}^2_\beta(1-2\epsilon)^{|\beta|}$$

The expectation in (3.3) is not only over the verifiers choice of f, g_1 and μ, but also over its choice of U, W and h. Thus, we have shown that

$$\mathrm{E}_{U,W,h}\left[\sum_\beta \hat{A}_{\pi_2(\beta)}\hat{B}^2_\beta(1-2\epsilon)^{|\beta|}\right] \;=\; \delta\ . \tag{3.5}$$

We now define the strategies in the two-prover game.

P_1: Receives W, and picks a $\beta \subseteq \{\pm 1\}^W$, such that each β is picked with probability $\hat{B}^2_\beta$. Then it returns a uniformly random $y \in \beta$.

P_2: Receives U, and picks a $\alpha \subseteq \{\pm 1\}^U$, such that each α is picked with probability $\hat{A}^2_\alpha$. Then it returns a uniformly random $z \in \alpha$.

According to (3.1) we have that $\hat{A}_\emptyset = \hat{B}_\emptyset = 0$, as A and B are folded over true. Thus, we do not need to worry that either α or β are empty sets. Parseval's identity (3.2) ensures that the sum of all probabilities equals to one. We note that if the proof is a correct $\mathrm{SWP}(u)$ for an assignment x, then $\hat{A}_{\{x|_U\}} = \hat{B}_{\{x|_W\}} = 1$. In that case P_1 returns $y = x|_W$ and P_2 returns $z = x|_U$.

The verifier accepts if and only if the values of z and y are consistent with each other. If $\alpha = \pi_2(\beta)$, then this probability is at least $|\beta|^{-1}$ because for every element $z \in \alpha$ there has to exist at least one element $y \in \beta$ such that $z = y|_U$. Thus, the probability that the verifier accepts in the two-prover game is at least

$$\sum_\beta \hat{A}^2_{\pi_2(\beta)} \hat{B}^2_\beta |\beta|^{-1} \ .$$

We use Cauchy-Schwartz' inequality, Proposition 2.1, and Parseval's identity (3.2) in order to relate an expression similar to the one in (3.5) with the success probability in the two-prover game.

$$\begin{aligned}\sum_\beta \hat{A}_{\pi_2(\beta)} \hat{B}^2_\beta |\beta|^{-1/2} &\le \left(\sum_\beta \hat{A}^2_{\pi_2(\beta)} \hat{B}^2_\beta |\beta|^{-1}\right)^{1/2} \left(\hat{B}^2_\beta\right)^{1/2} \\ &= \left(\sum_\beta \hat{A}^2_{\pi_2(\beta)} \hat{B}^2_\beta |\beta|^{-1}\right)^{1/2}\end{aligned}$$

We use this inequality and the fact that for a random variable X, we have that $\mathrm{E}\left[X^2\right] \ge \mathrm{E}\left[X\right]^2$:

$$\begin{aligned}\mathrm{E}_{U,W,h}\left[\sum_\beta \hat{A}^2_{\pi_2(\beta)} \hat{B}^2_\beta |\beta|^{-1}\right] &\ge \mathrm{E}_{U,W,h}\left[\left(\sum_\beta \hat{A}_{\pi_2(\beta)} \hat{B}^2_\beta |\beta|^{-1/2}\right)^2\right] \\ &\ge \left(\mathrm{E}_{U,W,h}\left[\sum_\beta \hat{A}_{\pi_2(\beta)} \hat{B}^2_\beta |\beta|^{-1/2}\right]\right)^2 \ . \qquad (3.6)\end{aligned}$$

The final step in relating the success probability of the verifier and (3.5) is to relate $|\beta|^{-1/2}$ with $(1 - 2\epsilon)$. The following lemma helps us to do this.

Lemma 3.9. *For* $1/2 > \epsilon > 0$ *and* $\beta \neq \emptyset$

$$|\beta|^{-1/2} \quad \geq \quad 2\sqrt{\epsilon}(1-2\epsilon)^{|\beta|} \ .$$

Proof. By the power series expansion of e we have that $e^x \geq 1 + x$ for any real x. We make use of this in the first and last inequality of the following calculations:

$$\begin{aligned} (4\epsilon|\beta|)^{-1} &\geq \frac{1}{e^{4\epsilon|\beta|}-1} \\ &\geq \frac{1}{e^{4\epsilon|\beta|}} \\ &= \left(e^{-2\epsilon}\right)^{2|\beta|} \\ &\geq (1-2\epsilon)^{2|\beta|} \ . \end{aligned}$$

Thus $(4\epsilon|\beta|)^{-1} \geq (1-2\epsilon)^{2|\beta|}$. Multiplying with 4ϵ and taking the square root of both sides concludes the proof. □

We apply Lemma 3.9 on the expression for the accept probability (3.6).

$$\begin{aligned} \mathrm{E}_{U,W,h}\left[\sum_{\beta} \hat{A}^2_{\pi_2(\beta)}\hat{B}^2_{\beta}|\beta|^{-1}\right] &\geq \left(\mathrm{E}_{U,W,h}\left[\sum_{\beta} \hat{A}_{\pi_2(\beta)}\hat{B}^2_{\beta}|\beta|^{-1/2}\right]\right)^2 \\ &\geq 4\epsilon\left(\mathrm{E}_{U,W,h}\left[\sum_{\beta} \hat{A}_{\pi_2(\beta)}\hat{B}^2_{\beta}(1-2\epsilon)^{|\beta|}\right]\right)^2 \\ &= 4\epsilon\delta^2 \end{aligned}$$

This concludes the proof of Lemma 3.8. □

The assumption of Lemma 3.8,

$$\mathrm{E}\left[A(f)B(g_1)B(g_2)\right] \quad = \quad \delta \ ,$$

implies that the probability that the verifier accepts is $(1+\delta)/2$. From Theorem 3.2 we have that the soundness of the u-parallel two-prover game is d_c^u, for some constant $d_c < 1$. By Lemma 3.8 we have that the soundness of the PCP is $(1+\sqrt{d_c^u/4\epsilon})/2$. For every $\epsilon > 0$ we can make the soundness of the PCP arbitrarily close to $1/2$ by choosing a large enough u.

3.5 Putting the Pieces Together

We are now ready to prove the main result of this chapter.

Theorem 3.10 (Håstad [22]). 3XOR *is approximation resistant.*

Proof. Given an arbitrary language L in NP, we have shown that for every $\epsilon > 0$ we can create a PCP verifier for L with acceptance condition 3XOR which has completeness $1 - \epsilon$ and soundness $1/2 + \epsilon$. Furthermore, the verifier only uses a logarithmic number of random bits. As $2^{-3}|\text{3XOR}^{-1}(1)| = 1/2$ we can apply Theorem 2.22 and conclude that 3XOR is approximation resistant.

□

Chapter 4

Non-Trivial Approximations of CSPs

In this chapter we describe a general technique for approximating Max CSP instances. It works by finding a solution that gives a positive contribution on the small degree terms of the Fourier spectra of the objective function, while seeing to that the higher degree terms give a smaller positive or negative contribution. Our focus is to outperform the random assignment algorithm if this is possible. By using our method we are able to show that a large number of predicates are not approximation resistant. For example, we show in Theorem 4.4 that if P is a predicate of arity $2s$ and has at most $2s + 1$ accepting inputs, then it is not approximation resistant. The focus in this chapter is on Max CSP(P), however the method is potentially applicable on instances of all kinds of Max CSPs.

4.1 Method

The objective function of a Max CSP(P) instance, I, is the weighted sum of the indicator variables of the constraints. In Section 2.4 we saw that each such term can be written as a multilinear expression. The whole sum can thus also be written as a multilinear expression

$$I(x_1, \ldots x_n) \;=\; \sum_{i=1}^{m} w_i P(z_{i1}, \ldots z_{ik}) \;=\; \sum_{S \subseteq [n], |S| \le k} c_S \prod_{i \in S} x_i \;. \tag{4.1}$$

In fact this is a Fourier spectra of the objective value function and the Fourier coefficients are given by

$$c_S \;=\; 2^{-n} \sum_{x \in \{\pm 1\}^n} I(x) \prod_{i \in S} x_i \;.$$

As all constraints in I depend on k variables we have that $c_S = 0$ for sets S with more than k elements. Thus in (4.1) we only sum over sets S where $|S| \le k$.

A random assignment assigns each binary variable x_i the value -1 or 1 randomly and independently. Such an assignment gives every term of (4.1), that includes at least one variable x_i, an expected value of zero. The only term that gives a contribution to the expectation is the constant term, $c_\emptyset$. In order to beat a random assignment we want to produce an assignment with an expected weight of at least $c_\emptyset + \gamma w_{\text{tot}}$, for a constant $\gamma > 0$.

If there is a large weight on the linear terms of the Fourier spectra, $\sum_{i=1}^{n} |c_{\{i\}}| = \delta w_{\text{tot}}$ for a constant $\delta > 0$, then it is easy to assign values to the variables such that the linear terms give a contribution of δw_{tot}. The problem is that there is no control on the higher order terms which possibly could give a negative contribution that cancel the contribution from the linear terms. In order to reduce the contribution of the higher order terms, we therefore only give a small bias towards the solution that maximize the linear terms. With probability ε, depending on k and δ, we assign x_i to $\operatorname{sgn}(c_{\{i\}})$, and otherwise assign x_i a random value. The linear terms then give an expected positive contribution of $\varepsilon\delta w_{\text{tot}}$ at the same time as the absolute contribution of each term of degree i is reduced with a factor of ε^i compared with the solution that maximized the linear terms. By choosing an appropriately small value on ε we can make the total absolute contribution of the higher degree terms at most one third of the contribution of the linear terms. The algorithm then finds an assignment with expected value of at least $c_\emptyset + 2\varepsilon\delta w_{\text{tot}}/3$, and thereby it beats a random assignment.

We can apply the same method on the bi-linear terms, $\sum_{i<j} c_{\{i,j\}} x_i x_j$. This is a Max 2AllEqual instance and we can use the Charikar-Wirth algorithm in order to find a good approximate solution, $(a_i)_{i=1}^{n}$. Negating the solution do not change the value on the bi-linear terms, we can therefore make the sum of the linear terms positive. We choose an appropriately small value on ε and for every variable x_i, with probability ε we assign it the same value as a_i, and otherwise a random unbiased value. By this procedure we ensure that the expected contribution of the higher degree terms is small in comparison with the expected contribution from the linear and bi-linear terms.

For some predicates P it is possible to show that an assignment that almost satisfies an instance of Max CSP(P) has to give a non-negligible positive value on either the linear or bi-linear terms of the Fourier spectra. In order to show that a predicate is not approximation resistant, it is enough to create an approximation algorithm that beats a random assignment on satisfiable and almost satisfiable instances. This is because a random assignment achieves the same expected objective value irrespective of the satisfiability of the instance. Thus, satisfiable and almost satisfiable instances are the ones where a random assignment achieves the worst approximation ratio. By combining the algorithm for almost satisfiable instances with a random assignment, we then get a better approximation ratio than using a random assignment alone. Thus, if an assignment that almost satisfies an instance of Max CSP(P) has to give a non-negligible positive value on either the linear or bi-linear terms this implies that the condition for our algorithm to outperform a random assignment is satisfied, which in turn shows that P is not approximation resistant.

4.2 Advantage on Linear and Bi-Linear Terms

As mentioned in the introduction, the objective function of a Max CSP(P) instance can be described as a multilinear expression, see (4.1). In this section we show that if there is an assignment that gives a non-negligible positive value to the linear or bi-linear terms of this expression, then we can do better than just picking a random assignment.

Linear Terms

Let us consider the linear terms of the objective value function, $I^{(1)}(x_1, \dots x_n) = \sum_{i=1}^{n} c_{\{i\}} x_i$. It is easy to see that the linear terms are maximized by assigning x_i to $\mathrm{sgn}(c_{\{i\}})$, giving the value $\sum_i |c_{\{i\}}|$. However, in order to control the expected value of higher degree terms Algorithm Lin picks an assignment that is ε-biased towards this assignment. The expected value of the terms of degree i decrease their value with a factor of ε^i. By choosing a small enough bias we ensure that the expected value of the non-constant terms is dominated by the linear terms. The following theorem quantifies the performance of Algorithm Lin.

```
Input: A Max CSP($P$) instance $I$
For $i := 1, \dots n$ do:
    • with probability $\varepsilon$: assign $x_i := \begin{cases} 1 & \text{if } c_{\{i\}} \geq 0 \\ -1 & \text{otherwise} \end{cases}$,
    • or else assign $x_i$ according to an unbiased coin flip.
```

Figure 4.1: Algorithm Lin - for approximating Max CSP(P).

Theorem 4.1. *Let $I(x_1, \dots x_n) = \sum_{S \subseteq [n], |S| \leq k} c_S \prod_{i \in S} x_i$ be the objective value function for an instance of* Max CSP(P)*, $P : \{\pm 1\}^k \to \{0, 1\}$. Denote the sum of all constraints' weights w_{tot}. If $\sum_i |c_{\{i\}}| \geq \delta w_{tot}$, for a constant $\delta > 0$, then Algorithm* Lin*, with $\varepsilon = \delta/2k$, produces a solution with expected weight of at least $c_\emptyset + \frac{\delta^2}{3k} w_{tot}$.*

Proof. The expected objective value is $\mathrm{E}[I] = \mathrm{E}\left[I^{(0)} + I^{(1)} + I^{(\geq 2)}\right]$. In order to get a lower bound on $\mathrm{E}[I]$, we first have

$$\mathrm{E}\left[I^{(1)}\right] = \varepsilon \sum_{i=1}^{n} |c_{\{i\}}| \geq \varepsilon \delta w_{\text{tot}} \ .$$

The following lemma will help in establishing a lower bound for $\mathrm{E}\left[I^{(\geq 2)}\right]$.

Lemma 4.2. *Let c_S be as in Theorem 4.1 and $j \geq 1$, then*

$$\sum_{S\subseteq[n],|S|=j} |c_S| \quad \leq \quad \frac{w_{tot}}{2}\binom{k}{j}^{1/2} \quad .$$

Proof. We know that $I(x_1, \ldots x_n)$ is a sum of weighted predicate indicator functions, $\sum_l w_l P(z_{l1}, \ldots z_{lk})$. Let $(c^l_S)_{S\subseteq[n]}$ be the Fourier coefficients of the l'th constraint,

$$P(z_{l1}, \ldots z_{lk}) = \sum_{S\subseteq[n],|S|\leq k} c^l_S \prod_{i\in S} x_i \quad .$$

We see that $c_S = \sum_l w_l c^l_S$. Note that almost all c^l_S are equal to zero. If for some $i \in S$, x_i does not appear as a part of the input $(z_{l1}, \ldots z_{lk})$, then $c^l_S = 0$. Thus, if $|S| = j$ there are at most $\binom{k}{j}$ non-zero c^l_S for a fixed value of l. The sum of all Fourier coefficients of degree j terms is now bounded using Cauchy-Schwartz inequality,

$$\begin{aligned} \sum_{S\subseteq[n],|S|=j} |c^l_S| \quad &\leq \quad \left(\sum_{S\subseteq[n]:c^l_S\neq 0,|S|=j} \left(c^l_S\right)^2\right)^{1/2} \left(\sum_{S\subseteq[n]:c^l_S\neq 0,|S|=j} 1\right)^{1/2} \\ &\leq \quad \left(\sum_{S\subseteq[n]:c^l_S\neq 0,S\neq\emptyset} \left(c^l_S\right)^2\right)^{1/2} \binom{k}{j}^{1/2} \quad . \end{aligned} \tag{4.2}$$

We analyze the first factor in (4.2) by using Parseval's identity and using the fact that P and $P - 1/2$ have the same Fourier coefficients except for $c^l_\emptyset$. We let $c'_\emptyset$ be the Fourier coefficient for $P - 1/2$.

$$\begin{aligned} \sum_{S\subseteq[n]:c^l_S\neq 0,S\neq\emptyset} \left(c^l_S\right)^2 \quad &\leq \quad \left(c'_\emptyset\right)^2 + \sum_{S\subseteq[n]:S\neq\emptyset} \left(c^l_S\right)^2 \\ &= \quad 2^{-n} \sum_{x\in\{\pm1\}^n} \left(P(z_{l1}, \ldots z_{lk}) - \frac{1}{2}\right)^2 \\ &= \quad \frac{1}{4} \end{aligned}$$

Putting this in (4.2) we obtain the following bound:

$$\sum_{S\subseteq[n],|S|=j} |c^l_S| \quad \leq \quad \frac{1}{2}\binom{k}{j}^{1/2} \quad .$$

By adding each constraint's Fourier coefficients we complete the proof of the lemma:

$$\begin{aligned}\sum_{S\subseteq[n],|S|=j} |c_S| &\leq \sum_{l,S\subseteq[n],|S|=j} w_l|c_S^l| \\ &\leq \sum_l w_l \frac{1}{2}\binom{k}{j}^{1/2} \\ &= \frac{w_{\text{tot}}}{2}\binom{k}{j}^{1/2} \quad .\end{aligned}$$

□

The expected value of a term of degree t is given by

$$\mathrm{E}\left[\prod_{j=1}^{t} x_{i_j}\right] = \varepsilon^t \prod_{j=1}^{t} \operatorname{sgn}(c_{i_j}) \quad .$$

To see this, note that if any of the variables $x_{i_1}, \ldots x_{i_t}$ is set according to an unbiased coin flip, then the expected value of the product is zero. The probability that none of the t variables is set to a random value is ε^t and in that case it is set to $\prod_{j=1}^{t} \operatorname{sgn}(c_{i_j})$. We can now give a lower bound of terms of degree ≥ 2:

$$\begin{aligned}\mathrm{E}\left[I^{(\geq 2)}\right] &\geq \sum_{j=2}^{k} -\varepsilon^j \frac{w_{\text{tot}}}{2}\binom{k}{j}^{1/2} \\ &\geq -\frac{w_{\text{tot}}}{2}\sum_{j=2}^{k} \varepsilon^j k^{j/2} \quad . \end{aligned} \tag{4.3}$$

We set $\varepsilon = \delta/2k$. Setting $j = 1$ in Lemma 4.2 we get the upper bound $\delta \leq \sqrt{k}/2$. The geometric sum in (4.3) can now be analyzed by noting that $\varepsilon\sqrt{k} \leq 1/4$.

$$\begin{aligned}\sum_{j=2}^{k} \varepsilon^j k^{j/2} &\leq \varepsilon^2 k \sum_{j=0}^{k-2} \left(\frac{1}{4}\right)^j \\ &< \frac{4}{3}\varepsilon^2 k \\ &= \frac{\delta^2}{3k} \quad .\end{aligned}$$

Substituting this in (4.3) we get

$$\mathrm{E}\left[I^{(\geq 2)}\right] \geq -\frac{\delta^2}{6k} w_{\text{tot}} \quad .$$

We note that $I^{(0)} = c_\emptyset$. Thus, we have

$$\begin{aligned} \mathrm{E}\left[I\right] &= \mathrm{E}\left[I^{(0)}\right] + \mathrm{E}\left[I^{(1)}\right] + \mathrm{E}\left[I^{(\geq 2)}\right] \\ &\geq c_\emptyset + \frac{\delta^2}{2k} w_{\mathrm{tot}} - \frac{\delta^2}{6k} w_{\mathrm{tot}} \\ &= c_\emptyset + \frac{\delta^2}{3k} w_{\mathrm{tot}} \ , \end{aligned}$$

which concludes the proof. □

Bi-Linear Terms

If there exists an assignment that makes the sum of the bi-linear terms non-negligibly positive, then Algorithm BiLin, defined in Figure 4.2, outperforms a random assignment.

Input: A Max CSP(P) instance I

1. Approximate the Max 2AllEqual instance, $I^{(2)}$, using the Charikar-Wirth algorithm [4] as explained in Section 2.7. Let $a_1, \ldots a_n$ be the produced solution.
2. If $I^{(1)}(a_1, \ldots a_n) < 0$ then do: $a_i := \overline{a}_i$ for $i := 1, \ldots n$.
3. Set $\alpha = I^{(2)}(a_1, \ldots a_n)/w_{\mathrm{tot}}$, where w_{tot} is the total weight in I. Set $\varepsilon = \max(\alpha/2k^{3/2}, 0)$, where k is the arity of P.
4. For $i := 1, \ldots n$ do:
    - with probability ε: assign $x_i := a_i$,
    - or else assign x_i according to an unbiased coin flip.

Figure 4.2: Algorithm BiLin - for approximating Max CSP(P).

Theorem 4.3. *Let $I(x_1, \ldots x_n) = \sum_{S \subseteq [n], |S| \leq k} c_S \prod_{i \in S} x_i$ be the objective value function for an instance of* Max CSP(P)*, $P : \{\pm 1\}^k \to \{0, 1\}$. Denote the sum of all constraints' weights w_{tot}. If there exists, for a constant $\delta > 0$, an assignment such that $\sum_{1 \leq i < j \leq n} c_{\{i,j\}} x_i x_j \geq \delta w_{tot}$, then Algorithm* BiLin *produces a solution with expected weight of at least $c_\emptyset + \kappa w_{tot}$, where $\kappa > 0$ and only depends on constants δ and k.*

Proof. Maximizing $I^{(2)}$ is a Max 2AllEqual instance. As a first step of Algorithm BiLin we run the Charikar-Wirth algorithm for Max 2AllEqual [4] trying to maximize $I^{(2)}$. The algorithm returns a solution $a_1, \ldots a_n$. Let

$$I^{(2)}(a_1, \ldots a_n) \quad = \quad \alpha w_{\mathrm{tot}} \ .$$

By Lemma 4.2, with $j = 2$, we have the following upper bound on $w_{\text{bi-lin}}$, the total weight in the Max 2AllEqual instance $I^{(2)}$:

$$w_{\text{bi-lin}} \leq \frac{w_{\text{tot}}}{2}\sqrt{\frac{k(k-1)}{2}} < \frac{w_{\text{tot}}k}{2} .$$

The optimal gain on $I^{(2)}$ is thus at least $2\delta/k$. By Lemma 2.23 we have a lower bound on the expected value of α over the random choices of the Charikar-Wirth algorithm:

$$\mathrm{E}\left[\alpha\right] \geq c_{\text{cw}}\delta/\log(2\delta/k)^{-1} ,$$

where c_{cw} is a positive constant from Lemma 2.23. If the linear terms give a negative contribution, $I^{(1)}(a_1, \ldots a_n) < 0$, then the solution $a_1, \ldots a_n$ is negated which does not effect $I^{(2)}$ but negates $I^{(1)}$, thus ensuring that the expected value of $I^{(1)}$ is non-negative.

The expected value of the sum of the bi-linear terms can be calculated by considering each term separately. Both variables in a term are set according to the solution $a_1, \ldots a_n$ with probability ε^2. This gives a contribution of $\varepsilon^2\alpha w_{\text{tot}}$. If any of the variables instead are set according to a coin flip, then the expected value is zero. Thus, we have $\mathrm{E}\left[I^{(2)}\right] = \varepsilon^2\alpha w_{\text{tot}}$. The high degree terms can be lower bounded by using a similar argument as in the proof of Theorem 4.1:

$$\begin{aligned}
\mathrm{E}\left[I^{(\geq 3)}\right] &\geq -\sum_{S\subseteq[n],3\leq|S|\leq k} \varepsilon^{|S|}|c_S| \\
&\geq -\sum_{j=3}^{k} \varepsilon^j \frac{w_{\text{tot}}}{2}\binom{k}{j}^{1/2} \\
&\geq -\frac{w_{\text{tot}}}{2}\sum_{j=3}^{k} \varepsilon^j k^{j/2} \\
&> -w_{\text{tot}}\varepsilon^3 k^{3/2} ,
\end{aligned}$$

where the last inequality holds if ε is set to a value such that $\varepsilon\sqrt{k} \leq 1/2$.

The expected objective value can now be bounded from below,

$$\begin{aligned}
\mathrm{E}\left[I\right] &= c_\emptyset + \mathrm{E}\left[I^{(1)} + I^{(2)} + I^{(\geq 3)}\right] \\
&\geq c_\emptyset + \left(\varepsilon^2\alpha - \varepsilon^3 k^{3/2}\right) w_{\text{tot}} . \qquad (4.4)
\end{aligned}$$

If $\alpha > 0$ we set $\varepsilon = \alpha/2k^{3/2}$, otherwise we set $\varepsilon = 0$. We need to ensure that $\varepsilon\sqrt{k} \leq 1/2$:

$$\begin{aligned}
\varepsilon\sqrt{k} &= \frac{\alpha}{2k} \\
&\leq \frac{1}{8} ,
\end{aligned}$$

where we used that $\alpha \leq k/4$ according to Lemma 4.2 with $j = 2$. The lower bound (4.4) ensures that $\mathrm{E}[I]$ is at least $c_\emptyset + \left(\frac{\alpha}{2k}\right)^3 w_{\text{tot}}$. Choosing $\kappa = \left(\frac{\alpha}{2k}\right)^3$, the theorem follows as the expected value of α is at least $c_{\text{cw}}\delta/\log(2\delta/k)^{-1}$ and $\left(\frac{\alpha}{2k}\right)^3$ is a convex function for non-negative α. □

4.3 Predicates of Arity k

As previously mentioned, in order to show that a predicate is not approximation resistant, it is sufficient to outperform a random assignment on satisfiable and almost satisfiable instances. This is because picking a random assignment achieves the worst approximation ratios for these instances. In this section, we show that for some predicates P, satisfiable and almost satisfiable assignments to Max CSP(P) instances must have some positive weight on either the linear or bi-linear terms of the instance. By applying the theorems of the previous section we conclude that such predicates are not approximation resistant. In particular, we show that predicates with few accepting inputs are not approximation resistant.

Theorem 4.4. *Let $P : \{\pm 1\}^k \to \{0, 1\}$, $k \geq 3$. If P has at most $2\lfloor k/2 \rfloor + 1$ accepting inputs, then P is not approximation resistant.*

Remark 4.5. Engebretsen and Holmerin [7] designed an approximation resistant predicate $P : \{\pm 1\}^6 \to \{0, 1\}$, with only eight accepting inputs. The above theorem is thus tight for the case $k = 6$ as it stipulates that if a predicate has at most seven accepting inputs, then it is not approximation resistant.

In order to prove Theorem 4.4 we need two lemmas. Lemma 4.6 shows that if $P^{(1)}(y) + P^{(2)}(y)$ is positive for all accepting inputs y, then P is not approximation resistant. Lemma 4.7 is used for obtaining lower bounds on $P^{(1)}(y) + P^{(2)}(y)$.

Lemma 4.6. *Let $P : \{\pm 1\}^k \to \{0, 1\}$ be a predicate. If $P^{(1)}(y) + P^{(2)}(y) > 0$ for all $y \in P^{-1}(1)$, then P is not approximation resistant.*

Proof. Let $\delta = \min_{y \in P^{-1}(1)} P^{(1)}(y) + P^{(2)}(y)$, where $\delta > 0$ according to the assumption in the lemma. Furthermore, define $\gamma = \min_{y \in \{\pm 1\}^k} P^{(1)}(y) + P^{(2)}(y)$. We know that $\gamma < 0$ as $\sum_{y \in \{\pm 1\}^k} P^{(1)}(y) + P^{(2)}(y) = 0$. Set $\varepsilon = \frac{\delta}{2(\delta - \gamma)}$. If there is no assignment that satisfies weight of at least $(1 - \varepsilon) w_{\text{tot}}$, then a random assignment achieves an approximation ratio larger than $2^{-k}|P^{-1}(1)|/(1 - \varepsilon) > (1 + \varepsilon)2^{-k}|P^{-1}(1)|$. Thus, the interesting case is if there is an assignment that satisfies weight at least $(1 - \varepsilon) w_{\text{tot}}$. Assume therefore that this is the case and calculate a lower bound on the linear and bi-linear terms of the objective value function,

$$\begin{aligned}
I^{(1)}(x_1,\ldots)+I^{(2)}(x_1,\ldots) &= \sum_{i=1}^{m} w_i\left(P^{(1)}(z_{i1},\ldots z_{ik})+P^{(2)}(z_{i1},\ldots z_{ik})\right)\\
&\geq \left(\left(1-\frac{\delta}{2(\delta-\gamma)}\right)\delta+\frac{\delta}{2(\delta-\gamma)}\gamma\right)w_{\text{tot}}\\
&= \left(\delta+\frac{\gamma-\delta}{2(\delta-\gamma)}\delta\right)w_{\text{tot}}\\
&= \frac{\delta}{2}w_{\text{tot}} \ .
\end{aligned}$$

Thus, we know that either $I^{(1)}(x_1,\ldots)\geq\frac{\delta}{4}w_{\text{tot}}$ or $I^{(2)}(x_1,\ldots)\geq\frac{\delta}{4}w_{\text{tot}}$. In the first case Theorem 4.1 implies that P is not approximation resistant and in the latter case Theorem 4.3 implies that P is not approximation resistant. □

Lemma 4.7. *Let $x,y\in\{\pm1\}^k$ and $d(x,y)$ be the Hamming distance between x and y. Define $t=d(x,y)-k/2$. Then the following holds:*

1. $\psi_x^{(1)}(y)=-2t\cdot2^{-k}$ *and*
2. $\psi_x^{(2)}(y)=(2t^2-k/2)2^{-k}$,

where $\psi_x(y)$ is defined in (2.1).

Proof. Let $z\in\{\pm1\}^k$ such that $z_i=x_iy_i$. The number of -1 in z equals the Hamming distance $d(x,y)$. Thus,

$$2^k\psi_x^{(1)}(y)=\sum_{i=1}^{k}x_iy_i=\sum_{i=1}^{k}z_i=k-2d(x,y)=-2t \ ,$$

establishing the first claim. The second claim is verified by simple calculation:

$$\begin{aligned}
2^k\psi_x^{(2)}(y) &= \sum_{1\leq i<j\leq k} z_iz_j\\
&= \sum_{\substack{i\,:\,z_i=1\\ j\,:\,z_j=1\\ i<j}} z_iz_j+\sum_{\substack{i\,:\,z_i=-1\\ j\,:\,z_j=-1\\ i<j}} z_iz_j+\sum_{\substack{i\,:\,z_i=1\\ j\,:\,z_j=-1}} z_iz_j\\
&= \binom{k/2-t}{2}+\binom{k/2+t}{2}-\left(\frac{k}{2}+t\right)\left(\frac{k}{2}-t\right)\\
&= 2t^2-\frac{k}{2} \ .
\end{aligned}$$

□

Proof of Theorem 4.4. We use (2.2) to express the linear and bi-linear terms of a constraint $P(y)$,

$$P^{(1)}(y) + P^{(2)}(y) = \sum_{\alpha \in P^{-1}(1)} \psi_\alpha^{(1)}(y) + \psi_\alpha^{(2)}(y) \ .$$

Lemma 4.7 give the following lower bound for each term in the above expression,

$$\begin{aligned} \psi_\alpha^{(1)}(y) + \psi_\alpha^{(2)}(y) &= (-2t + 2t^2 - k/2)2^{-k} \\ &\geq \begin{cases} -\frac{k}{2}2^{-k} & \text{if } k \text{ even} \\ -\frac{k+1}{2}2^{-k} & \text{if } k \text{ odd} \end{cases} , \end{aligned} \tag{4.5}$$

where $t = d(x,y) - k/2$, as defined in Lemma 4.7. In the above calculation we used the fact that $-2t+2t^2 \geq 0$ if k is even (equality when $t = 0, 1$) and $-2t+2t^2 \geq -1/2$ if k is odd (equality when $t = 1/2$). If $\alpha = y$, we have that $\psi_\alpha^{(1)}(y) + \psi_\alpha^{(2)}(y) = \frac{(k+1)k}{2}2^{-k}$. Thus, if $y \in P^{-1}(1)$, the following lower bound is derived by applying (4.5)

$$\begin{aligned} P^{(1)}(y) + P^{(2)}(y) &= \sum_{\alpha \in P^{-1}(1)} \psi_\alpha^{(1)}(y) + \psi_\alpha^{(2)}(y) \\ &\geq \left(\frac{(k+1)k}{2} - (s-1) \left\lceil \frac{k}{2} \right\rceil \right) 2^{-k} \\ &\geq \frac{k}{2}2^{-k} \begin{cases} \text{if } s \leq k+1 \text{ and } k \text{ even} \\ \text{if } s \leq k \text{ and } k \text{ odd} \end{cases} , \end{aligned}$$

where s is the number of accepting inputs to P. Applying Lemma 4.6 concludes the proof of Theorem 4.4. □

4.4 Generalizing the Method

In Lemma 4.6 we showed that if $P^{(1)}(y)+P^{(2)}(y)$ is strictly positive for all accepting inputs y to a predicate P, then P is not approximation resistant. In this section, we show we can instead consider the sum $C \cdot P^{(1)}(y) + P^{(2)}(y)$, for an arbitrary constant C, and also allow at most two accepting inputs that make the sum equal to zero, instead of positive, and P can still be shown to not be approximation resistant. This is expressed in Theorem 4.9. That theorem relies on the following rather technical generalization of Lemma 4.6.

Lemma 4.8. *Let $P : \{\pm 1\}^k \to \{0,1\}$ be a predicate. An instance of* Max CSP(P) *has the objective value function $I(x_1, \ldots x_n) = \sum_{i=1}^m w_i P(z_{i1}, \ldots z_{ik})$. For an assignment to $x_1, \ldots x_n$, let u_y be the weight on input y. For constants C and $\delta > 0$, there exist constants $\varepsilon > 0$ and $\kappa > 0$, such that if $\sum_{y \in P^{-1}(1)} u_y(C \cdot P^{(1)}(y) + P^{(2)}(y)) \geq \delta w_{tot}$, where $w_{tot} = \sum_{i=1}^m w_i$, then either the value of the assignment is less than $(1-\varepsilon)w_{tot}$, or Algorithm* Lin *or Algorithm* BiLin *will achieve an approximation ratio of at least $2^{-k}|P^{-1}(1)| + \kappa$.*

Proof. Using the assumptions in the lemma we conclude that

$$\begin{aligned} C \cdot I^{(1)}(x_1, \ldots) + I^{(2)}(x_1, \ldots) &= \sum_{y \in \{\pm 1\}^k} u_y (C \cdot P^{(1)}(y) + P^{(2)}(y)) \\ &\geq \delta w_{\text{tot}} + \sum_{y \in P^{-1}(0)} u_y (C \cdot P^{(1)}(y) + P^{(2)}(y)) \ . \end{aligned}$$

Negating an assignment negates the value of $I^{(1)}$ but does not effect $I^{(2)}$. Thus, if $C < 0$, then the assignment can be negated and we set $C := -C$, ensuring that C is non-negative. From now on we assume C to be non-negative. The terms of the last expression are bounded from below using Lemma 4.7,

$$\begin{aligned} C \cdot P^{(1)}(y) + P^{(2)}(y) &= C \cdot \sum_{\alpha \in P^{-1}(1)} \psi_\alpha^{(1)}(y) + \sum_{\alpha \in P^{-1}(1)} \psi_\alpha^{(2)}(y) \\ &\geq C \left(-k 2^{-k}\right) |P^{-1}(1)| + \left(-(k/2) 2^{-k}\right) |P^{-1}(1)| \\ &\geq -Ck - k/2 \\ &> -(C+1)k \ . \end{aligned}$$

By setting $\varepsilon = \frac{\delta}{2(C+1)k}$, we ensure that the weight on non-accepting inputs is at most $\frac{\delta}{2(C+1)k} w_{\text{tot}}$, implying that

$$\begin{aligned} C \cdot I^{(1)}(x_1, \ldots) + I^{(2)}(x_1, \ldots) &\geq \delta w_{\text{tot}} - \sum_{y \in P^{-1}(0)} u_y (C+1)k \\ &\geq \frac{\delta}{2} w_{\text{tot}} \ . \end{aligned}$$

Thus, either $I^{(1)}(x_1, \ldots) \geq \frac{\delta}{4C} w_{\text{tot}}$ or $I^{(2)}(x_1, \ldots) \geq \frac{\delta}{4} w_{\text{tot}}$. Applying Theorems 4.1 and 4.3 concludes the proof. □

Theorem 4.9. *Let $P : \{\pm 1\}^k \to \{0, 1\}$ be a predicate. If for some constant C, $C \cdot P^{(1)}(y) + P^{(2)}(y)$ is zero for at most two $y \in P^{-1}(1)$ and strictly positive for all other $y \in P^{-1}(1)$, then P is not approximation resistant.*

Before we prove Theorem 4.9 we prove the following lemma which relies on Zwick's algorithm for almost satisfiable 2SAT instances [43].

Lemma 4.10. *Let P be a predicate with exactly two accepting inputs. If there exists a $(1 - \varepsilon)$-satisfying solution to a* Max CSP(P) *instance, then a $(1 - O(\varepsilon^{1/3}))$-satisfying solution can be produced in probabilistical polynomial time.*

Proof. Assume that $a = (a_1, \ldots a_k)$, $b = (b_1, \ldots b_k) \in \{\pm 1\}^k$ are the two accepting inputs to P. We show that there exists a set of 2SAT clauses such that only a and b satisfy all of them. The set is simply the maximal set of clauses of size two that satisfies both a and b. Let us call this set D.

Claim 4.11. *Only a and b satisfies all the* 2SAT *clauses in D.*

Proof of Claim 4.11. Let $c \in \{\pm 1\}^k$ such that $c \neq a$, $c \neq b$. We show that there exists a clause in D that c does not satisfy. As $a \neq b$ there exists an $i \in [k]$ such that $a_i \neq b_i$. Without loss of generality we assume that $c_i = a_i$. As $a \neq c$ we also know that $a_j \neq c_j$ for some $j \in [k]$. The 2SAT clause over x_i and x_j that rejects c accepts both a and b because $a_j \neq c_j$ and $b_i \neq c_i$. This clause is therefore contained in D. □

Each constraint in the original instance is transformed into the above set of 2SAT clauses. If an assignment satisfies an original constraint then it also satisfies all corresponding 2SAT clauses. However, if it does not satisfy an original constraint then at least one of the 2SAT clauses is not satisfied by that assignment. According to the assumption in the lemma we have that an optimal solution to the Max CSP(P) instance is at least $(1-\varepsilon)$-satisfying. This implies that the same assignment also satisfies at least a $1-\varepsilon$ fraction of the total weight in the resulting Max 2SAT instance. By using Zwick's algorithm for almost satisfiable 2SAT instances, Theorem 2.24, we then find a $1-O(\varepsilon^{1/3})$ satisfying solution. Each P-constraint is transformed into a constant number of 2SAT clauses, thus the solution also $1-O(\varepsilon^{1/3})$ satisfies the original Max CSP(P) instance. This concludes the proof of Lemma 4.10. □

Proof of Theorem 4.9. Let I be an instance of Max CSP(P) with total weight w_{tot} and let y_1 and y_2 be the accepting inputs to P that make $C \cdot P^{(1)}(y) + P^{(2)}(y) = 0$. If there is only one such input, let y_1 be that input and y_2 be an arbitrary accepting input that is not equal to y_1. If no accepting inputs to P make $C \cdot P^{(1)}(y) + P^{(2)}(y) = 0$, then let y_1 and y_2 be distinct arbitrary accepting inputs to P. For an optimal assignment to I, let u_y be the weight of an input y. For any constant value of $\varepsilon > 0$, we define three non-exclusive cases:

- The value of an optimal solution is less than $(1-\varepsilon)w_{\text{tot}}$.
- The weight on y_1 and y_2 is large, namely $u_{y_1} + u_{y_2} \geq (1-2\varepsilon)w_{\text{tot}}$.
- For $\delta = \varepsilon \min_{y \in P^{-1}(1) \setminus \{y_1, y_2\}} \{C \cdot P^{(1)}(y) + P^{(2)}(y)\}$,

$$\sum_{y \in P^{-1}(1)} u_y (C \cdot P^{(1)}(y) + P^{(2)}(y)) \geq \delta w_{\text{tot}} \ .$$

We first show that every instance is covered by at least one of the above cases. Then we show that in each case we can achieve an approximation ratio of at least $2^{-k}|P^{-1}(1)| + \kappa$, for a constant $\kappa > 0$ as long as ε is small enough. This shows that P is not approximation resistant.

We assume that none of the first two cases apply on an instance and show that this implies that the last case is applicable. The first case implies that an optimal solution has value $w_{\text{opt}} = \sum_{y \in P^{-1}(1)} u_y \geq (1-\varepsilon)w_{\text{tot}}$. The second case implies that $u_{y_1} + u_{y_2} < (1-2\varepsilon)w_{\text{tot}}$. Thus we have

$$\sum_{y \in P^{-1}(1) \setminus \{y_1, y_2\}} u_y > \varepsilon w_{\text{tot}} \ .$$

We calculate a lower bound for the sum specified in the third case.

$$\begin{aligned}\sum_{y \in P^{-1}(1)} u_y(C \cdot P^{(1)}(y) + P^{(2)}(y)) \quad &\geq \quad \sum_{y \in P^{-1}(1) \setminus \{y_1, y_2\}} u_y(C \cdot P^{(1)}(y) + P^{(2)}(y)) \\ &\geq \quad \sum_{y \in P^{-1}(1) \setminus \{y_1, y_2\}} u_y \delta / \varepsilon \\ &> \quad \delta w_{\text{tot}}\end{aligned}$$

This implies that the third case is applicable.

If an instance is in the first case, a random assignment approximates the instance well enough with an approximation ratio of

$$\frac{2^{-k}|P^{-1}(1)|w_{\text{tot}}}{(1-\varepsilon)w_{\text{tot}}} \quad > \quad 2^{-k}(1+\varepsilon)|P^{-1}(1)| \ .$$

If it is in the second case we apply Lemma 4.10 with the predicate that only accepts y_1 and y_2. If it is in the third case we instead apply Lemma 4.8.

□

Chapter 5

Approximation Resistant Predicates

In this chapter we study approximation resistant predicates. First we extend a result by Samorodnitsky and Trevisan [35] in order to show a large number of predicates to be approximation resistant. A consequence of that result is that all predicates with few non-accepting inputs are approximation resistant.

The method described in Chapter 4 gives interesting implications for optimal solutions to hard instances of some Max CSPs. For example, Håstad produced almost satisfiable and satisfiable instances of Max E3SAT such that it is NP-hard to find a solution that is essentially better than the expected performance of a random assignment. The proposed satisfying assignment made either one or all three of the literals true for almost all constraints. We show that this is no coincidence. In fact, if there is an almost satisfying assignment that on a constant fraction of the constraints make exactly two literals true, then the instance can be approximated better than using a random assignment. This gives us some new interesting insights about the hardness of approximating Max E3SAT.

5.1 The Samorodnitsky-Trevisan Predicate

A Samorodnitsky-Trevisan predicate $P_{ST^{s,t}} : \{\pm 1\}^{s+t+st} \to \{0,1\}$ consists of st different parity checks, each acting on three variables,

$$P_{ST^{s,t}}(x_1, \dots x_s, x'_1, \dots x'_t, x''_1, \dots x''_{st}) \quad = \quad \bigwedge_{\substack{1 \le i \le s \\ 1 \le j \le t}} x_i \oplus x'_j \oplus x''_{(i-1)s+j} \ .$$

It is easy to see that if an input x is accepted by $P_{ST^{s,t}}$, then the first $s+t$ bits determine the other st bits in x. Thus, there are only 2^{s+t} accepting inputs out of a total of 2^{s+t+st} inputs.

Samorodnitsky and Trevisan [35] exhibit a non-adaptive PCP for every language in NP with completeness $1-\epsilon$, soundness $2^{-st}+\epsilon$ and $q = s+t+st$ query bits, for any constant $\epsilon > 0$. Furthermore, the verifier uses $P_{ST^{s,t}}$ as acceptance condition. We have that Theorem 2.22 implies the following theorem.

Theorem 5.1 (Samorodnitsky-Trevisan [35]). *$P_{ST^{s,t}}$ is approximation resistant.*

A predicate P is said to be hereditary approximation resistant if all predicates that are implied by P are approximation resistant. The following theorem states that Samorodnitsky-Trevisan predicates are hereditary approximation resistant.

Theorem 5.2. *Let $P : \{\pm 1\}^{s+t+st} \to \{0,1\}$ be a predicate that is implied by $P_{ST^{s,t}}$, i.e., $P^{-1}_{ST^{s,t}}(1) \subseteq P^{-1}(1)$. Then P is approximation resistant.*

5.2 Proof of Theorem 5.2

We prove Theorem 5.2 by constructing an efficient PCP having a verifier with acceptance condition P. We then apply Theorem 2.22 in order to conclude that P is approximation resistant. This is basically the same method as was used in Chapter 3 in order to show that 3XOR is approximation resistant. In this proof we rely heavily on the soundness analysis of the Samorodnitsky-Trevisan PCP made by Håstad and Wigderson [24].

The Verifier

The proof used is a standard written proof, SWP(u), the same as is used in Chapter 3. It consists of supposed long code encodings of possible answers in the parallel two-prover game, which are subsets of an assignment x to a SAT formula ϕ. However, the verifier checks the proof somewhat differently. The verifier inspects $s + t + st$ bits in the proof and decides whether or not to accept by applying the predicate P on the (possibly negated) bits in the proof. The actions of the verifier are described in Figure 5.1 and we call this Test ST^P.

A very similar verifier is defined in [24]. The only difference that we make is in the accept criterion. Instead of using P it uses the Samorodnitsky-Trevisan predicate $P_{ST^{s,t}}$ in order to decide whether to accept or reject. We call this Test ST and it is shown in Figure 5.2.

The accept criterion of the verifier in Test ST can be seen as a conjunction of st parity tests:

$$A(f_i)B_j(g_j)B_j(g_{ij}) \quad = \quad 1 \ , \ \ i \in [s], j \in [t] \ .$$

If the SWP(u) is a correct proof in the sense of Definition 3.5, then each such test is passed with probability $1 - \epsilon$. This is because $A(f_i)B_j(g_j)B_j(g_{ij}) = \mu_{ij}(x|_{W_j})$ as $A(f_i) = f_i(x|_U)$, $B_j(g_j) = g_j(x|_{W_j})$ and $B_j(g_{ij}) = f_i(x|_U)g_j(x|_{W_j})\mu_{ij}(x|_{W_j})$. The tests are independent and thus the completeness is $(1-\epsilon)^{st}$. The completeness of Test ST^P is at least as high as in Test ST, as the accept criterion is more accepting. Thus the completeness of Test ST^P is at least $(1-\epsilon)^{st}$.

Input: A SWP(u).

1. The verifier chooses a set U of u variables and s random boolean functions f_i, $i = 1, 2, \ldots s$ on U. Let A be the portion of the proof corresponding to U.
2. For $j = 1, 2, \ldots t$ the verifier repeats the following steps. For each variable in U choose a random clause containing it. Let h_j be the conjunction of the chosen clauses and let W_j be the set of variables appearing in the chosen clauses. Choose g_j to be a random boolean function on W_j. Let B_j be the portion of the proof corresponding to W_j, but folded over true and conditioned upon h_j.
3. For $i = 1, 2, \ldots s$, $j = 1, 2, \ldots t$ choose a function μ_{ij} on W_j which, independently at each point takes the value 1 with probability $1 - \epsilon$ and -1 with probability ϵ. Set $g_{ij} = g_i f_i \mu_{ij}$, i.e., $g_{ij}(y) = f_i(y|_U) g_j(y) \mu_{ij}(y)$.
4. The verifier accepts if

$$P\left(-A(f_1), \ldots - A(f_s), B_1(g_1), \ldots B_t(g_t), B_1(g_{11}), \ldots B_t(g_{st})\right) \quad ,$$

otherwise it rejects.

Figure 5.1: Test ST^P

Input: A SWP(u).
Steps 1 to 3 are performed as in Test ST^P.

4 The verifier accepts if

$$P_{ST^{s,t}}\left(-A(f_1), \ldots - A(f_s), B_1(g_1), \ldots B_t(g_t), B_1(g_{11}), \ldots B_t(g_{st})\right) \quad ,$$

otherwise it rejects.

Figure 5.2: Test ST

The Soundness of the PCP

A verifier with acceptance condition P accepts a completely random proof with probability $2^{-(s+t+st)}|P^{-1}(1)|$. This probability is thus a lower bound for the soundness.

In Test ST, the verifier accepts if $A(f_i)B_j(g_j)B_j(g_{ij}) = 1$ for $1 \leq i \leq s$ and

$1 \le j \le t$. Thus, the following expression is 1 if the verifier accepts and 0 otherwise.

$$\prod_{\substack{1\le i\le s\\1\le j\le t}} \frac{1 + A(f_i)B_j(g_j)B_j(g_{ij})}{2}$$

Calculating the above product we get the sum

$$2^{-st} \sum_{S\subseteq[s]\times[t]} \prod_{(i,j)\in S} A(f_i)B_j(g_j)B_j(g_{ij}) \ .$$

If ϕ is at most c-satisfiable, then this expression is bounded by the following lemma as the soundness of the u-parallel two-prover game can be made arbitrarily small by choosing u large enough.

Lemma 5.3 (Håstad-Wigderson [24]). *Suppose $S \subseteq [s] \times [t]$ and non-empty. Let*

$$\mathrm{E}\left[\prod_{(i,j)\in S} A(f_i)B_j(g_j)B_j(g_{ij})\right] \quad = \quad \delta \ ,$$

where the expectation is taken over all coin tosses of the PCP *verifier. Then there is a strategy for the two provers in the parallel two-prover game that convinces the verifier with probability at least $4\epsilon\delta^2$.*

The accept indicator function for Test ST^P is

$$\sum_{\substack{(a_1,\ldots b_1,\ldots b_{11},\ldots)\\ \in P^{-1}(1)}} \left(\prod_{i=1}^{s} \frac{1+a_iA(f_i)}{2}\right)\left(\prod_{j=1}^{t} \frac{1+b_jB_j(g_j)}{2}\right)\left(\prod_{\substack{i\in[s]\\j\in[t]}} \frac{1+b_{ij}B_j(g_{ij})}{2}\right), \tag{5.1}$$

We analyze (5.1) by showing that the terms which neither are constants or on the form $\prod_{(i,j)\in S} A(f_i)B_j(g_j)B_j(g_{ij})$ have a zero expectation. Ignoring the sign, an arbitrary term can be written on the following form

$$2^{-(s+t+st)} \prod_{i\in I} A(f_i) \prod_{j\in J} B_j(g_j) \prod_{(i,j)\in S} B_j(g_{ij}) \ ,$$

where $I \subseteq [s]$, $J \subseteq [t]$ and $S \subseteq [s] \times [t]$. If we cannot use Lemma 5.3 in order to bound a non-constant term, we know that there is an i_0 (or a j_0) such that if $i_0 \in I$ (or $j_0 \in J$) then there is an even number of elements in $\{(i_0, j) \in S\}$ (or in $\{(i, j_0) \in S\}$) and if $i_0 \notin I$ (or $j_0 \notin J$) then there is an odd number of elements in $\{(i_0, j) \in S\}$ (or in $\{(i, j_0) \in S\}$). Assume for now that there is such an i_0 and that $i_0 \in I$. The term can now be written as

$$\left(A(f_{i_0}) \prod_{(i_0,j)\in S} B_j(g_{i_0j})\right) C \ , \tag{5.2}$$

where the value of C is independent of the choices of f_{i_0}, $g_{i_0 1}$, $\dots$ $g_{i_0 t}$. Let us fix the choices of g_1, $\dots$ g_t. It is equally probable for a verifier to choose a function f_{i_0} as it is to choose $-f_{i_0}$. Thus the (partial) choice of the following functions $(f_{i_0}, g_1, \dots g_t, g_{i_0 1}, \dots g_{i_0 t})$ is as frequent as $(-f_{i_0}, g_1, \dots g_t, -g_{i_0 1}, \dots - g_{i_0 t})$. Due to folding of the long code tables we know that $A(f_{i_0}) = -A(-f_{i_0})$. Furthermore, we know that the number of elements in $\{(i_0, j) \in S\}$ is even, thus

$$\begin{aligned}\prod_{(i_0,j)\in S} B_j(g_{i_0 j}) &= \prod_{(i_0,j)\in S} -B_j(-g_{i_0 j}) \\ &= (-1)^{|\{(i_0,j)\in S\}|} \prod_{(i_0,j)\in S} B_j(-g_{i_0 j}) \\ &= \prod_{(i_0,j)\in S} B_j(-g_{i_0 j}) \ .\end{aligned}$$

We conclude that

$$A(f_{i_0}) \prod_{(i_0,j)\in S} B_j(g_{i_0 j}) = -A(-f_{i_0}) \prod_{(i_0,j)\in S} B_j(-g_{i_0 j}) \ ,$$

and that the expectation of expression (5.2) must be zero. If $i_0 \notin I$ or there is a j_0 such that $|\{j \in J : j = j_0\}|$ does not have the same parity as $|\{(i,j) \in S : j = j_0\}|$, then the same line of reasoning shows that the expectation of the term is zero in those cases as well.

The constant term in the accept probability (5.1) is $2^{-(s+t+st)}|P^{-1}(1)|$, which also is the value of the largest possible coefficient for a term in general. Theorem 3.2 ensures that the soundness in the u-parallel game is at most d_c^u, for some constant $d_c < 1$. The expected value of $\prod_{(i,j)\in S} A(f_i)B_j(g_j)B_j(g_{ij})$, for any set $S \subseteq [s] \times [t]$, is upper bounded by $\sqrt{d_c^u/4\epsilon}$. This is seen by combining Lemma 5.3 and Theorem 3.2. There are $2^{st} - 1$ different nonempty sets S. All other terms in (5.1) have been shown to have an expected value of 0. Thus the soundness of the PCP is upper bounded by

$$\begin{aligned}&2^{-(s+t+st)}|P^{-1}(1)| + (2^{st} - 1)2^{-(s+t+st)}|P^{-1}(1)|\sqrt{d_c^u/4\epsilon} \leq \\ &\qquad 2^{-(s+t+st)}|P^{-1}(1)| + 2^{-(s+t)}|P^{-1}(1)|\sqrt{d_c^u/4\epsilon} \ .\end{aligned}$$

For any constant $\epsilon' > 0$ we can, by choosing a large enough u in the parallel two prover game, get a soundness for Test ST^P of $2^{-(s+t+st)}|P^{-1}(1)| + \epsilon'$.

P is Approximation Resistant

Theorem 3.1 gives a polynomial transformation ϕ that given a language $L \in \mathrm{NP}$ and an element v produces a E3CNF(5)-formula $\phi(v)$ such that:

- $\phi(v)$ is satisfiable if $v \in L$, and

- $\phi(v)$ is not c-satisfiable if $v \notin L$.

Our PCP distinguishes between these two cases. The proof is a standard written proof of $\phi(v)$ and the verifier has acceptance condition P, uses logarithmic (in $|v|$) number of random bits. Furthermore, for every constant δ, we can make the soundness less than $2^{-(s+t+st)}|P^{-1}(1)| + \delta$ by choosing a u large enough in the standard written proof, and we can make the completeness at least $1 - \delta$ by setting $\epsilon = \delta/st$. Thus, we have an efficient PCP construction for an arbitrary language in NP. We apply Theorem 2.22 and conclude that P is approximation resistant.

5.3 Predicates with Few Non-Accepting Inputs

If a predicate P has few non-accepting inputs, then there exists at least one predicate of the same type as P, such that it is implied by a Samorodnitsky-Trevisan predicate. Using Theorem 5.2, we then conclude that P is approximation resistant.

Theorem 5.4. *Let $k \geq s + t + st$ and $P : \{\pm 1\}^k \to \{0, 1\}$ be a predicate with at most $2^{st} - 1$ non-accepting inputs, then P is approximation resistant.*

Proof. Our goal is to design st special parity relations such that no non-accepting input satisfies all of them. Based on these parity relations and the predicate P, we construct the predicate P' which is implied by the Samorodnitsky-Trevisan predicate $P_{ST^{s,t}}$, and is of the same type as P. Using Theorem 5.2 we see that P' is approximation resistant and thus P is approximation resistant as well, because P and P' are of the same type.

Let $x \in \{\pm 1\}^k$ be a possible input to P, and $par_{ij}(x)$, where $i \in [s]$ and $j \in [t]$, is the parity check $x_i x_{s+j} x_{t+js+i}$. Initially, let $T := P^{-1}(0)$ be all non-accepting inputs of P. For each $(i, j) \in [s] \times [t]$ we repeat the following: Construct $T_{-1} = \{x \in T : par_{ij}(x) = -1\}$ and $T_1 = \{x \in T : par_{ij}(x) = 1\}$. If T_{-1} has fewer elements than T_1, set $T := T_{-1}$ and $b_{ij} := 1$, otherwise set $T := T_1$ and $b_{ij} := -1$. After each iteration, T is at most half of its previous size. The assumption of the theorem stipulates that T initially consists of at most $2^{st} - 1$ elements. Thus, after st iterations T must be the empty set.

We get P' by applying a bitmask on st bits of the input to P:

$$P'(x_1, x_2, \ldots x_k) \quad = \quad P(x_1, \ldots x_{s+t}, b_{11}x_{s+t+1}, \ldots b_{st}x_{s+t+st}, x_{s+t+st+1}, \ldots x_k) \ ,$$

for all inputs $(x_1, \ldots x_k) \in \{\pm 1\}^k$. As we have only negated some input bits on certain positions it is clear that P and P' are of the same type and thus Max CSP(P) and Max CSP(P') are equivalent problems. Furthermore, P' is implied by $P_{ST^{s,t}}$ because of the construction of the bitmask values $\{b_{ij}\}$. Theorem 5.2 then shows that P' is approximation resistant, and thereby also P. We note that if $k > s + t + st$, then $P_{ST^{s,t}}$ is considered to take inputs of length k but the last $k - (s + t + st)$ input bits will never effect the output of the predicate. □

In the proof of Theorem 5.4 we used the fact that a bitmask to the input does not change the type of the predicate. Clearly we could also use the fact that permuting the input bits does not change the type of the predicate either. Unfortunately, this approach makes the analysis much more complex and it would not strengthen Theorem 5.4 significantly. However, for predicates of arity five we are able to use this approach in order to produce a tight result. Theorem 5.4, with $s = 1$ and $t = 2$, lets us conclude that if a predicate P of arity five has at most three non-accepting inputs, then P is approximation resistant. We better this by the following theorem.

Theorem 5.5. *Let P be a predicate of arity 5 with at most 5 non-accepting inputs, then P is approximation resistant.*

Proof. Let $P^{-1}(0) = \{s_1, s_2, s_3, s_4, s_5\}$ where $s_j \in \{\pm 1\}^5$ for $1 \le j \le 5$. With $s_j[i]$ we denote the i'th bit in s_j. We start by describing a sufficient condition for P being approximation resistant.

Let a parity test $\{t, u, v\} \subset [5]$ on s be defined as $\prod_{i \in \{t,u,v\}} s[i]$. A pair of parity tests is admissible if the intersection of them contains a single element. The result of an admissible pair of parity tests is in $\{\pm 1\}^2$, thus there are four possible outcomes.

Claim 5.6. *Assume there exists an admissible pair of parity tests such that there is a possible outcome that do not occur if applied on all elements in $P^{-1}(0)$. Then P is approximation resistant.*

Proof of Claim 5.6. Let the admissible pair of parity checks be $\{t, u, v\}$ and $\{t, w, z\}$ and the outcome that not occurs (b_1, b_2). Define P' such that

$$P(x_1, x_2, x_3, x_4, x_5) = P'(x_t, -b_1 x_u, -b_2 x_w, x_v, x_z) \ .$$

P' is apparently of the same type as P. In addition P' is implied by $P_{ST^{1,2}}$, as

$$P_{ST^{1,2}}(x_1, x_2, x_3, x_4, x_5) = (x_1 \oplus x_2 \oplus x_4) \wedge (x_1 \oplus x_3 \oplus x_5) \ .$$

Theorem 5.2 shows that P' is approximation resistant and thus P is approximation resistant as well. □

To prove the theorem we assume that there do not exist any admissible pairs of parity checks as specified in the claim, and show that this leads to a contradiction. Therefore, assume that there exists $P^{-1}(0) = \{s_1, s_2, s_3, s_4, s_5\}$ such that, for every possible choice of admissible pair of parity checks, par_1 and par_2, for every possible outcome $(b_1, b_2) \in \{\pm 1\}^2$ of par_1 and par_2, there is at least one element $s_j \in P^{-1}(0)$ such that $(par_1(s_j), par_2(s_j)) = (b_1, b_2)$.

If not every single parity check on three variables will split $\{s_1, s_2, s_3, s_4, s_5\}$ into sets of size two and three, then it is easy to see that the above assumption is false. Thus, we can assume that every parity check splits $\{s_1, s_2, s_3, s_4, s_5\}$ into sets of sizes two and three. Now, w.l.o.g. we assume that

$$\prod_{i=1}^{3} s_1[i] = \prod_{i=1}^{3} s_2[i] \neq \prod_{i=1}^{3} s_3[i] = \prod_{i=1}^{3} s_4[i] = \prod_{i=1}^{3} s_5[i] \ , \tag{5.3}$$

because we can rearrange the values of $\{s_i\}$. Let par_1 be the parity check on variables $(1, 2, 3)$. Now there are three different possible parity checks for par_2, on variables $(1, 4, 5)$, $(2, 4, 5)$ or $(3, 4, 5)$. Each possible choice of par_2 will have to give different output when applied on s_1 and s_2.

We know that $s_1[i] = s_2[i]$ for some $1 \leq i \leq 3$ because of the first parity check. It can be seen that $(s_1[1], s_1[2], s_1[3]) = (s_2[1], s_2[2], s_2[3])$, because otherwise at least one of the possible parity checks will give the same answer for both s_1 and s_2. To see this, let $s_1[t] = s_2[t]$ and $s_1[u] \neq s_2[u]$ for $t, u \in \{1, 2, 3\}$. If $s_1[4]s_1[5] = s_2[4]s_2[5]$, then

$$\prod_{i \in \{t,4,5\}} s_1[i] = \prod_{i \in \{t,4,5\}} s_2[i]$$

and else

$$\prod_{i \in \{u,4,5\}} s_1[i] = \prod_{i \in \{u,4,5\}} s_2[i] \ .$$

Furthermore, $s_1[4] \oplus s_1[5] \neq s_2[4] \oplus s_2[5]$ because otherwise any one of the parity checks would yield the same result for s_1 and s_2. Thus, s_1 and s_2 only differ on either the fourth or the fifth element. We assume w.l.o.g. that s_1 and s_2 differ only on the last element.

We study what happens if we choose par_1 differently. Instead of $(1, 2, 3)$ we let par_1 check parity on variables $(3, 4, 5)$. This check separates s_1 from s_2. We assume w.l.o.g. that

$$\prod_{i=3}^{5} s_1[i] = \prod_{i=3}^{5} s_3[i] \neq \prod_{i=3}^{5} s_2[i] = \prod_{i=3}^{5} s_4[i] = \prod_{i=3}^{5} s_5[i] \ . \tag{5.4}$$

Reapplying the same arguments as above we conclude that $(s_1[3], s_1[4], s_1[5]) = (s_3[3], s_3[4], s_3[5])$ and that $s_1[1]s_1[2] \neq s_3[1]s_3[2]$. Assume w.l.o.g. that $s_1[1] \neq s_3[1]$ and $s_1[2] = s_3[2]$.

Let par_1 now act on the variables $(2, 3, 4)$. All these bits are identical in s_1, s_2 and s_3, thus

$$\prod_{i=2}^{4} s_4[i] = \prod_{i=2}^{4} s_5[i] \neq \prod_{i=2}^{4} s_1[i] = \prod_{i=2}^{4} s_2[i] = \prod_{i=2}^{4} s_3[i] \ .$$

Now we see that $(s_4[2], s_4[3], s_4[4]) = (s_5[2], s_5[3], s_5[4])$ and $s_4[1]s_4[5] \neq s_5[1]s_5[5]$. We conclude that either $\prod_{i=1}^{3} s_4[i] \neq \prod_{i=1}^{3} s_5[i]$ or $\prod_{i=3}^{5} s_4[i] \neq \prod_{i=3}^{5} s_5[i]$. But this contradicts (5.3) or (5.4). Thus the assumption of Claim 5.6 is satisfied and we can conclude that P is approximation resistant which proves the theorem.

Obviously, if P has fewer than five accepting inputs, then an admissible pair of parity checks exist as well and P is approximation resistant in this case as well. $\square$

Remark 5.7. In Chapter 6 we show that there are predicates of arity four, having exactly three non-accepting inputs, which are not approximation resistant. This implies that there are also predicates of arity five, having exactly six non-accepting inputs, which are not approximation resistant. Thus, Theorem 5.5 is optimal in this sense.

5.4 Characterization of Hard Instances

A hard Max CSP instance is an instance that is satisfiable or almost satisfiable and for which it is NP-hard to approximate it significantly better than picking a random assignment. In this section we characterize optimal and near-optimal solutions to such instances by bounding the weight of particular inputs for such solutions. Our main tool for this task is Lemma 4.8.

Zwick characterized all predicates of arity three [42]. From this work we know that approximation resistant predicates of arity three are of the same type as either 3XOR which accepts if and only if the parity of the input bits is odd, 3NTW which accepts if and only if not exactly two of the input bits are true, $\text{3OXR}(x_1, x_2, x_3) = x_1 \vee (x_2 \oplus x_3)$, or $\text{3OR}(x_1, x_2, x_3) = x_1 \vee x_2 \vee x_3$.

Theorem 5.8. *Let P be* 3NTW, 3OXR *or* 3OR, *and let I be an instance of* Max CSP(P). *For every $\gamma > 0$, there exist $\varepsilon > 0$ and $\kappa > 0$ such that if there exists a $(1-\varepsilon)$-satisfying assignment with at least weight $\gamma w_{\text{tot}}(I)$ on even parity inputs, then I can be $(s2^{-3} + \kappa)$-approximated in probabilistical polynomial time, where s is the number of accepting inputs to P.*

Proof. The multilinear expression for odd parity is

$$\text{3XOR}(x_1, x_2, x_3) = \frac{4 - 4x_1x_2x_3}{8} \; .$$

For P, there are $s - 4$ accepting inputs with even parity. Using (2.2), it can be concluded that $P^{(0)}(x_1, x_2, x_3) = \frac{s}{8}$ and $P^{(3)}(x_1, x_2, x_3) = \frac{s-8}{8} x_1x_2x_3$. If (x_1, x_2, x_3) has odd parity, then $P^{(0)}(x_1, x_2, x_3) + P^{(3)}(x_1, x_2, x_3) = 1$ implying that $P^{(1)}(x_1, x_2, x_3) + P^{(2)}(x_1, x_2, x_3) = 0$. If $(x_1, x_2, x_3) \in P^{-1}(1)$ but have even parity, then $P^{(0)}(x_1, x_2, x_3) + P^{(3)}(x_1, x_2, x_3) = \frac{2s}{8} - 1$ implying that $P^{(1)}(x_1, x_2, x_3) + P^{(2)}(x_1, x_2, x_3) = 2 - \frac{2s}{8} > 0$. The theorem now follows from applying Lemma 4.8. $\square$

The theorem shows that it is no coincidence that the PCP verifier of Håstad [22], that asks three questions, almost always accepts a correct proof due to parity.

In fact, assume that a PCP verifier can express any language in NP by asking for three bits and having almost completeness one and soundness only negligibly higher than the acceptance probability of a random proof. Then Theorem 5.8 implies that a correct proof will almost always be accepted due to the parity of the three bits asked for.

There are some additional observations that can be made about 3OR. The sums of the linear and bi-linear terms are for all 3OR's accepting inputs listed in Table 5.1.

Table 5.1: Values of sums of linear and bi-linear terms for the predicate 3OR.

x_1	x_2	x_3	$2^3 \cdot 3\mathrm{OR}^{(1)}$	$2^3 \cdot 3\mathrm{OR}^{(2)}$
1	1	−1	−1	1
1	−1	1	−1	1
1	−1	−1	1	1
−1	1	1	−1	1
−1	1	−1	1	1
−1	−1	1	1	1
−1	−1	−1	3	−3

Instead of looking at the sum of the linear and bi-linear terms we can look at only the bi-linear terms, or give more influence to the linear terms. By doing this we can deduce that an optimal solution to a hard instance of MAX CSP(3OR) must have roughly 1/4 of the weight on input $(-1,-1,-1)$. We have the following theorem.

Theorem 5.9. *Let I be an instance of* MAX CSP(3OR)*. For every $\gamma > 0$, there exist $\varepsilon > 0$ and $\kappa > 0$ such that if there exists a $(1-\varepsilon)$-satisfying assignment with weight $u_{(-1,-1,-1)}$ on input $(-1,-1,-1)$, and either $u_{(-1,-1,-1)} \le (1/4 - \gamma)w_{tot}(I)$ or $u_{(-1,-1,-1)} \ge (1/4+\gamma)w_{tot}(I)$, then I can be $(7/8+\kappa)$-approximated in probabilistical polynomial time.*

Proof. First we consider the case $u_{(-1,-1,-1)} \le (1/4-\gamma)w_{\mathrm{tot}}(I)$. We use Lemma 4.8 with $P = 3\mathrm{OR}$, $C = 0$ and $\delta = 3\gamma/8$. We calculate the sum specified in Lemma 4.8 with $C = 0$:

$$\begin{aligned}\sum_{y\in P^{-1}(1)} u_y P^{(2)}(y) &= -\frac{3}{8}u_{(-1,-1,-1)} + \sum_{y\in P^{-1}(1)\setminus\{(-1,-1,-1)\}} \frac{1}{8}u_y \\ &\ge -\frac{3}{8}(1/4-\gamma)w_{\mathrm{tot}}(I) + \frac{1}{8}(3/4)w_{\mathrm{tot}}(I) \\ &= (3\gamma/8)w_{\mathrm{tot}}(I)\ .\end{aligned}$$

In the inequality we assume that $\varepsilon \le \gamma$ and thus $\sum_{y\in P^{-1}(1)\setminus\{(-1,-1,-1)\}} u_y \ge 3/4$. By Lemma 4.8 we know that there exist positive constants ε' and κ such that either the value of an optimal assignment is less than $(1-\varepsilon)w_{\text{tot}}(I)$, or we can $(7/8+\kappa)$-approximate it. We set $\varepsilon = \min(\varepsilon', \gamma)$.

For the second case, $u_{(-1,-1,-1)} \geq (1/4+\gamma)w_{\text{tot}}(I)$, we use Lemma 4.8 with $P = 3\text{OR}$, $C = 2$ and $\delta = \gamma/2$. We calculate the sum specified in Lemma 4.8 with $C = 2$:

$$\begin{aligned}\sum_{y \in P^{-1}(1)} u_y(2 \cdot P^{(1)}(y) + P^{(2)}(y)) &\geq \frac{3}{8}u_{(-1,-1,-1)} + \sum_{y \in P^{-1}(1) \setminus \{(-1,-1,-1)\}} -\frac{1}{8}u_y \\ &\geq \frac{3}{8}(1/4+\gamma)w_{\text{tot}}(I) - \frac{1}{8}(3/4-\gamma)w_{\text{tot}}(I) \\ &= (\gamma/2)w_{\text{tot}}(I)\end{aligned}$$

By Lemma 4.8 we know that either the value of the assignment is less than $(1-\varepsilon)w_{\text{tot}}(I)$, or we can $(7/8+\kappa)$-approximate it.

□

We can make similar observations about the GLST predicate which is defined as

$$\text{GLST}(x_1, x_2, x_3, x_4) = (\overline{x}_1 \wedge (x_2 \equiv x_3)) \vee (x_1 \wedge (x_2 \equiv x_4)) \ . \tag{5.5}$$

This predicate was shown to be approximation resistant in [15]. The sums of the linear and bi-linear terms are for all accepting inputs listed in Table 5.2. Four of them have a positive value on the bi-linear terms and the remaining four have value zero.

Table 5.2: Values of sums of linear and bi-linear terms for the predicate GLST.

x_1	x_2	x_3	x_4	$\text{GLST}^{(1)}$	$\text{GLST}^{(2)}$
1	1	1	1	0	$1/2$
1	1	1	-1	0	0
1	-1	-1	1	0	0
1	-1	-1	-1	0	$1/2$
-1	1	1	1	0	$1/2$
-1	1	-1	1	0	0
-1	-1	1	-1	0	0
-1	-1	-1	-1	0	$1/2$

Using Lemma 4.8 we can prove the following theorem implying that optimal solutions to hard instances of Max CSP(GLST) have almost all weight on inputs $(1,1,1,-1)$, $(1,-1,-1,1)$, $(-1,1,-1,1)$ and $(-1,-1,1,-1)$.

Theorem 5.10. *Let I be an instance of* Max CSP(GLST). *For every $\gamma > 0$, there exist $\varepsilon > 0$ and $\kappa > 0$ such that if there exists a $(1-\varepsilon)$-satisfying assignment with total weight $\gamma w_{tot}(I)$ on inputs*

$$\{(1,1,1,1),(1,-1,-1,-1),(-1,1,1,1),(-1,-1,-1,-1)\} ,$$

then I can be $(1/2+\kappa)$-approximated in probabilistical polynomial time.

Proof. We use Lemma 4.8 with $P = \text{GLST}$, $C = 0$ and $\delta = \gamma/2$. We calculate the sum specified in Lemma 4.8 and use that $\text{GLST}^{(2)}$ equals $1/2$ for the inputs specified in the theorem. For all other satisfying inputs it is equal to 0.

$$\sum_{y \in P^{-1}(1)} u_y P^{(2)}(y) \quad = \quad (\gamma/2) w_{\text{tot}}(I)$$

By Lemma 4.8 we know that either the value of the assignment is less than $(1 - \varepsilon)w_{\text{tot}}(I)$, or we can $(1/2+\kappa)$-approximate it. $\square$

Chapter 6

Predicates of Arity Four

In this chapter we take a closer look at predicates of arity four. For each such predicate we try to deduce whether it is approximation resistant or not. There are $2^4 = 16$ different inputs and thus there are $2^{16} = 65536$ different predicates $P : \{\pm 1\}^4 \to \{0, 1\}$. Many of these are of the same type and thus have the same approximability. There are 2^4 bitmasks and 4! permutations that can be applied to the input bits. The largest number of predicates of the same type is thus $4! \cdot 2^4 = 384$. By using a simple computer program we see that there exist 402 different predicate types, including the two trivial types containing the constant predicates $P = 0$ and $P = 1$ respectively.

Goemans and Williamson [14] showed that all predicates depending on exactly two boolean variables are non-trivially approximable. Of course, this result also applies to predicates of arity four as long as the predicate only depends on exactly two of the input bits. Håstad [22] showed that if a predicate depends on three binary inputs and accepts all odd parity inputs or all even parity inputs, then it is approximation resistant. All other predicates, depending on three binary inputs, were shown to be non-trivially approximable by Zwick [42]. The situation for predicates of arity four is much less clear. Here we try to shed some light on the approximability of these predicates.

We apply and extend the PCP-techniques of [22] in order to show predicates to be approximation resistant. We also show non-trivial approximation algorithms for a large number of predicates based on the method in Chapter 4. Of the 400 non-trivial predicate types of arity four we show that 79 of these are approximation resistant while 275 are non-trivially approximable. In particular, all predicates with at most six accepting inputs are not approximation resistant. For every number of accepting inputs, the number of predicate types that have been characterized as non-trivially approximable and approximation resistant, respectively, are shown in Table 6.1. The chapter is concluded with tables showing, for each predicate type, its status in our classification.

Table 6.1: Approximability of MAX CSP(P) for predicates of arity four.

# accepting inputs	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	all
# non-trivially approx	1	4	6	19	27	50	50	52	27	26	9	3	1	0	0	275
# approx resistant	0	0	0	0	0	0	0	16	6	22	11	15	4	4	1	79
# unknown	0	0	0	0	0	0	6	6	23	2	7	1	1	0	0	46

6.1 Approximation Resistant Predicates

Håstad showed that if a predicate P of arity k accepts all inputs of even parity, then P is approximation resistant as long as $k \geq 3$.

Theorem 6.1 (Håstad [22]). *Let P be a predicate on k bits where $k \geq 3$ such that $P(x_1, \dots x_k) = 1$ for any $(x_i)_{i=1}^k$ satisfying $\prod_{i=1}^k x_i = 1$, then P is approximation resistant.*

Thus, all predicates implied by parity on four variables is approximation resistant. In the following four theorems we extend the methods in [22] which enables us to classify more predicates of arity four as approximation resistant.

Theorem 6.2. *Let P be a predicate of arity four implied by parity on the first three input bits, i.e., $P(x_1, x_2, x_3, x_4) = 1$ for any $(x_i)_{i=1}^4$ satisfying $3\text{XOR}(x_1, x_2, x_3) = 1$. Then P is approximation resistant.*

Proof. We show the theorem by producing a gadget that transforms a parity constraint $3\text{XOR}(x_1, x_2, x_3)$ into eight P-constraints:

$$\begin{array}{ll} P(x_1, x_2, x_3, y) & P(x_1, x_2, x_3, \overline{y}) \\ P(\overline{x}_1, \overline{x}_2, x_3, y) & P(\overline{x}_1, \overline{x}_2, x_3, \overline{y}) \\ P(\overline{x}_1, x_2, \overline{x}_3, y) & P(\overline{x}_1, x_2, \overline{x}_3, \overline{y}) \\ P(x_1, \overline{x}_2, \overline{x}_3, y) & P(x_1, \overline{x}_2, \overline{x}_3, \overline{y}) \ , \end{array}$$

where y is an auxiliary variable. Let s be the number of accepting inputs to P. Eight of these have odd parity on the first three bits and thus $s-8$ have even parity. The first three input bits to all constraints in the gadget have the same parity and all inputs are different. Thus, if $3\text{XOR}(x_1, x_2, x_3) = 1$, then all eight constraints are satisfied and else $s-8$ constraints are satisfied regardless of the value of y.

Let $\alpha = 8/(16-s)$ and assign each constraint in the gadget weight $w = 1/(16-s)$. If $3\text{XOR}(x_1, x_2, x_3) = 1$, then constraints of weight $8w = \alpha$ are satisfied and otherwise constraints of weight $(s-8)w = \alpha - 1$ are satisfied. Thus, the above is an α-gadget. By applying Lemma 2.25 we conclude that it is NP-hard to approximate MAX CSP(P) within $s/16+\epsilon$, for any $\epsilon > 0$. Thus, P is approximation resistant. □

Theorems 6.3 and 6.4 give sufficient conditions for a predicate to be approximation resistant. As in the proof of Theorem 6.2, we construct gadgets in order to prove Theorems 6.3 and 6.4. However, in these cases it does not work to do a simple local analysis of the gadgets, but instead we need to consider the whole instance constructed by the gadgets. Similar ideas about global analysis of gadgets were used by Feige and Reichman [11].

Theorem 6.3. *Let P be a predicate of arity four such that*

$$\sum_{(a_1,a_2,a_3,a_4)\in P^{-1}(1)} a_4 \quad = \quad 0$$

and $\{(a_1,a_2,a_3,1) : \text{3XOR}(a_1,a_2,a_3)=1\} \subset P^{-1}(1)$. *Then P is approximation resistant.*

Proof. We first construct a gadget that in conjunction with Lemma 2.25 would show P to be approximation resistant. However, the gadget is non-valid in our setting in that it uses constants so we need to construct a workaround.

We let s be the number of accepting inputs to P. Our non-valid gadget from $\text{3XOR}(x_1,x_2,x_3)$ to P is the following:

$$\begin{aligned}&P(x_1,x_2,x_3,1)\\&P(\overline{x}_1,\overline{x}_2,x_3,1)\\&P(\overline{x}_1,x_2,\overline{x}_3,1)\\&P(x_1,\overline{x}_2,\overline{x}_3,1)\ .\end{aligned}$$

The above is a $8/(16-s)$-gadget and by using Lemma 2.25 we could conclude that P is approximation resistant as was done in the proof of the previous theorem. However, the gadget is non-valid because in our setting we do not allow constants in the P-constraints as this would potentially change the performance of a random assignment.

We work around this problem by changing the constant 1 in the above gadget to an auxiliary variable u that is shared among all applications of the gadget. The crucial observation is now that there always exists an optimal solution to the Max CSP(P) instance that assigns $u = 1$. This global observation ensures that we can assume that u is set to 1 when making the local analysis of the gadget. However, the details need to be sorted out. We essentially follow the proof of Lemma 2.25.

We make the following observation: For an assignment with $u = -1$, either setting $u = 1$ or negating the whole assignment including u produces a solution with at least the same objective value.

To see this, we let t be the number of accepting inputs (a_1,a_2,a_3,a_4) such that $\text{3XOR}(a_1,a_2,a_3) = 1$ and $a_4 = -1$. We know that $s/2-4 \le t \le 4$. The condition of the theorem then give that there are $s/2-t$ accepting inputs such that $\text{3XOR}(a_1,a_2,a_3) = 0$ and $a_4 = -1$. We look at a the constraints generated from a gadget application on $3\mathrm{XOR}(x_1, x_2, x_3)$:

$$\begin{array}{l} P(x_1, x_2, x_3, u) \\ P(\overline{x}_1, \overline{x}_2, x_3, u) \\ P(\overline{x}_1, x_2, \overline{x}_3, u) \\ P(x_1, \overline{x}_2, \overline{x}_3, u) \ . \end{array}$$

If u is set to -1 and $3\mathrm{XOR}(x_1, x_2, x_3) = 1$ for an assignment, then exactly t of the four constraints are satisfied. If instead $3\mathrm{XOR}(x_1, x_2, x_3) = 0$, then $s/2 - t$ of the constraints are satisfied. Let an assignment y to the original Max CSP(3XOR) instance have value w, and the total weight of the instance be 1. If u is set to -1, then this assignment satisfies weight

$$tw + (s/2 - t)(1 - w) \tag{6.1}$$

in the corresponding Max CSP(P) instance where each constraint has the same weight as the constraint it originates from. Consider the solution that sets $u = 1$ and if $w < 1/2$ negates y and otherwise uses y. That assignment satisfies weight of

$$4 \max(w, 1 - w) + (s/2 - 4) \min(w, 1 - w) \ . \tag{6.2}$$

As $t \leq 4$ and $s/2 - t \leq 4$ we see that (6.2) is at least as large as (6.1). Thus, we conclude that there always exists an optimal solution to the Max CSP(P) instance that assigns $u = 1$.

Assume that the total weight of a Max CSP(3XOR) instance is 1. A solution that satisfies constraints of weight w then satisfies constraints of weight $4w + (s/2 - 4)(1 - w)$ in the transformed problem assuming u is set to 1 and each constraint in the gadget has the same weight as the constraint it originates from. For any constant $\delta > 0$ it is NP-hard to distinguish between an instance with an optimal solution satisfying constraints of weight $w = 1 - \delta$ and an instance with an optimal solution satisfying constraints of weight $w = 1/2 + \delta$ [22]. Using the above observation that there always exists an optimal solution assigning $u = 1$, we conclude that it is NP-hard to approximate Max CSP(P) better than

$$\frac{4(1/2 + \delta) + (s/2 - 4)(1/2 - \delta)}{4(1 - \delta) + (s/2 - 4)\delta} \ .$$

For any constant $\epsilon > 0$, we can choose a small enough δ such that the above expression is less than $s/16 + \epsilon$. Thus P is approximation resistant. □

Theorem 6.4. *Let P be a predicate of arity four such that*

$$\sum_{(a_1, a_2, a_3, a_4) \in P^{-1}(1)} a_1 a_4 \quad = \quad 0$$

and $\{(a_1, a_2, a_3, a_1) : 3\mathrm{XOR}(a_1, a_2, a_3) = 1\} \subset P^{-1}(1)$. *Then P is approximation resistant.*

Proof. We prove this theorem using the same technique as in the proof of Theorem 6.3. Once again we would like to use a non-valid gadget from $3\mathrm{XOR}(x_1, x_2, x_3)$ to P:

$$
\begin{aligned}
&P(x_1, x_2, x_3, x_1)\\
&P(\overline{x}_1, \overline{x}_2, x_3, \overline{x}_1)\\
&P(\overline{x}_1, x_2, \overline{x}_3, \overline{x}_1)\\
&P(x_1, \overline{x}_2, \overline{x}_3, x_1)\ .
\end{aligned}
$$

This time it is not allowed because a constraint may not act over the same variable multiple times. We work around this problem in the following manner: For each variable x_j occurring at the first position in a constraint we introduce an auxiliary variable x'_j. If z_{i_1} is x_j or $\overline{x}_j$, then z'_{i_1} denotes x'_j or $\overline{x'}_j$ respectively. For each constraint $3\mathrm{XOR}(z_{i_1}, z_{i_2}, z_{i_3})$ in the Max CSP(3XOR) instance we introduce four constraints:

$$
\begin{aligned}
&P(z_{i_1}, z_{i_2}, z_{i_3}, z'_{i_1})\\
&P(\overline{z}_{i_1}, \overline{z}_{i_2}, z_{i_3}, \overline{z'}_{i_1})\\
&P(\overline{z}_{i_1}, z_{i_2}, \overline{z}_{i_3}, \overline{z'}_{i_1})\\
&P(z_{i_1}, \overline{z}_{i_2}, \overline{z}_{i_3}, z'_{i_1})\ .
\end{aligned}
$$

We apply this gadget to Håstad's NP-hard Max CSP(3XOR) instances [22]. These instances have the property that all variables occurring at the first position in a constraint, only occur at that position in all other constraints as well. It is also, for any $\delta > 0$, NP-hard to distinguish whether an instance either is $1 - \delta$ satisfiable or not even $1/2 + \delta$ satisfiable. By setting x'_j to the same value as x_j is assigned, we get a very similar analysis as in the proof of Theorem 6.3. We let s be the number of accepting inputs to P. By using the conditions on P stipulated by the theorem we see that if $3\mathrm{XOR}(z_{i_1}, z_{i_2}, z_{i_3}) = 1$ and $z_{i_1} = z'_{i_1}$, then all 4 of the gadget's constraints are satisfied while if $3\mathrm{XOR}(z_{i_1}, z_{i_2}, z_{i_3}) = 0$ and $z_{i_1} = z'_{i_1}$ then $s/2 - 4$ constraints are satisfied. We let $s/2 - 4 \le t \le 4$ be the number of accepting inputs $(a_1, a_2, a_3, \overline{a}_1)$ such that $3\mathrm{XOR}(a_1, a_2, a_3) = 1$. If $z_{i_1} \ne z'_{i_1}$ and $3\mathrm{XOR}(a_1, a_2, a_3) = 1$, then t of the gadget's constraints are satisfied while if $z_{i_1} \ne z'_{i_1}$ and $3\mathrm{XOR}(a_1, a_2, a_3) = 0$, then $s/2 - t$ constraints are satisfied.

Assume that the total weight of a Max CSP(3XOR) instance is 1. A solution that satisfies constraints of weight w then satisfies constraints of weight $4w + (s/2 - 4)(1 - w)$ in the transformed problem if x'_j is set to the same value as x_j for all auxiliary variables. In order for the analysis in the proof of Theorem 6.3 to apply in this case, we need to show that we can assume that $x'_j = x_j$ in an optimal solution.

Given an arbitrary assignment to the Max CSP(P) instance we show that it is possible to produce an assignment with at least the same value that assigns $x'_j = x_j$. Assume that $x'_j \ne x_j$ in an assignment to a Max CSP(P) instance generated from gadget applications on one of Håstad's NP-hard Max CSP(3XOR) instance. Let the total weight of $3\text{XOR}(z_{i_1}, z_{i_2}, z_{i_3})$ constraints, where the literal z_{i_1} contains the variable x_j, be 1. Furthermore, let w' be the total weight of such constraints that are satisfied by the assignment. The weight of the corresponding constraints in the Max CSP(P) instance that are satisfied is then the same as in (6.1),

$$tw' + (s/2 - t)(1 - w') \ .$$

If $w' < 1/2$ we negate the value of x_j and otherwise we negate the value of x'_j, ensuring that x_j and x'_j are assigned the same value. The weight of the constraints that are satisfied is then the same as in (6.2),

$$4 \max(w', 1 - w') + (s/2 - 4) \min(w', 1 - w') \ .$$

Thus, the value of the solution has not decreased by the negation.

We apply the same argument as in the proof of Theorem 6.3. Given one of Håstad's NP-hard Max CSP(3XOR) instances I with total weight 1 we apply the gadget construction and get a Max CSP(P) instance I'. If I is $(1 - \delta)$ satisfiable, then there exists a solution to I' with value $4(1 - \delta) + (s/2 - 4)\delta$. If I is not $(1/2 + \delta)$ satisfiable, then there does not exist a solution to I' with value at least $4(1/2 + \delta) + (s/2 - 4)(1/2 - \delta)$. Thus, for any constant $\delta > 0$, it is NP-hard to achieve the approximation ratio

$$\frac{4(1/2 + \delta) + (s/2 - 4)(1/2 - \delta)}{4(1 - \delta) + (s/2 - 4)\delta}$$

for Max CSP(P). For any $\epsilon > 0$, we can choose a small enough δ such that the above expression is less than $s/16 + \epsilon$. Thus P is approximation resistant. □

Theorem 6.5. *Let P be a predicate such that*

$$\sum_{(a_1,a_2,a_3,a_4) \in P^{-1}(1)} a_3 a_4 \ \leq \ 0$$

and $\{(1,1,1,1), (1,-1,-1,-1), (-1,1,-1,-1), (-1,-1,1,1)\} \subset P^{-1}(1)$. *Then P is approximation resistant.*

Proof. The proof method is similar to the one used in proving Theorem 3.10. We build a PCP verifier with acceptance condition P. The completeness of the PCP is almost perfect and the soundness can be made arbitrarily close to the probability that a random proof is accepted. Creating a proof to the PCP such that the acceptance probability of the verifier is maximized can be considered as a Max CSP(P) problem. This is utilized by Theorem 2.22 that lets us conclude that P is approximation resistant. The actions of the PCP verifier is shown in Figure 6.1

A correct proof is a SWP(u) for some assignment x to a 3SAT formula. As

$$\{(1,1,1,1), (1,-1,-1,-1), (-1,1,-1,-1), (-1,-1,1,1)\} \subset P^{-1}(1) \ ,$$

Input: A SWP(u).

1. The verifier chooses a set U of u variables and a random boolean function f on U. Let A be the portion of the proof corresponding to U. A is folded over true.

2. For each variable in U choose a random clause containing it. Let h be the conjunction of the chosen clauses and let W be the set of variables appearing in the chosen clauses. Choose g_1 to be a random boolean function on W. Let B be the portion of the proof corresponding to W. B is folded over true and conditioned upon h.

3. Choose two functions μ and μ' on W by setting $\mu(y) = 1$ with probability $1 - \epsilon$ and $\mu(y) = -1$ otherwise, independently for each $y \in \{\pm 1\}^W$. Choose μ' independently from the same distribution as μ.

4. Set $g_2 = fg_1\mu$, i.e., define g_2 by for each $y \in \{\pm 1\}^W$, $g_2(y) = f(y|_U)g_1(y)\mu(y)$. Set $g_3 = fg_1\mu'$.

5. The verifier accepts if and only if $P(A(f), B(g_1), B(g_2), B(g_3))$.

Figure 6.1: A PCP verifier

we know that if $A(f)B(g_1)B(g_2) = 1$ and $A(f)B(g_1)B(g_3) = 1$, then the verifier accepts. If A and B are correct long code tables of $x|_U$ and $x|_W$, then $A(f) = f(x|_U)$, $B(g_1) = g_1(x|_W)$, $B(g_2) = f(x|_U)g_1(x|_W)\mu(x|_W)$ and $B(g_3) = f(x|_U)g_1(x|_W)\mu'(x|_W)$. This implies that $A(f)B(g_1)B(g_2) = \mu(x|_W)$ and that $A(f)B(g_1)B(g_3) = \mu'(x|_W)$. The probability that $\mu(x|_W) \neq 1$ or that $\mu'(x|_W) \neq 1$ is equal to $(1-\epsilon)^2$. Thus, a verifier accepts a correct proof with probability at least $(1-\epsilon)^2$.

As in Lemma 3.8 we prove the soundness by relating it to the soundness of the u-parallel two-prover game in Section 3.2.

Lemma 6.6. *Assume*

$$\mathrm{E}\left[P(A(f), B(g_1), B(g_2), B(g_3))\right] = c_\emptyset + \delta \ , \tag{6.3}$$

where the expectation is taken over all coin tosses of the PCP *verifier and* $c_\emptyset$ *is the fraction of accepting inputs to* P. *Then there is a strategy for the two provers in the two-prover game that convinces the verifier with probability at least* $4\epsilon\delta^2$.

Proof. We consider the multilinear expression for P:

$$P(x_1, x_2, x_3, x_4) = \sum_{S \subseteq [4]} c_S \prod_{i \in S} x_i \ ,$$

where

$$c_S \quad = \quad 2^{-4} \sum_{(x_1,x_2,x_3,x_4)\in\{\pm1\}^4} P(x_1,x_2,x_3,x_4)\prod_{i\in S} x_i \ . \tag{6.4}$$

The acceptance probability of the verifier is

$$\begin{aligned}&\mathrm{E}_{f,g_1,g_2,g_3}\left[P(A(f),B(g_1),B(g_2),B(g_3))\right] = \\ &c_\emptyset + c_{\{1\}}\mathrm{E}\left[A(f)\right] + c_{\{2\}}\mathrm{E}\left[B(g_1)\right] + \ldots + c_{\{1,2,3,4\}}\mathrm{E}\left[A(f)B(g_1)B(g_2)B(g_3)\right] \ .\end{aligned}$$

We start by showing that most of the terms in the above expression actually are zero. A vital observation is that A and B are folded, thus $A(f) = -A(-f)$ has to be true for any proof.

The functions f, g_1, g_2 and g_3 are chosen uniformly. Thus, the verifier chooses f and $-f$ with the same probability. Due to folding we then know that $\mathrm{E}_f\left[A(f)\right] = 0$. By the same argument we have that

$$\mathrm{E}_{g_1}\left[B(g_1)\right] = \mathrm{E}_{g_2}\left[B(g_2)\right] = \mathrm{E}_{g_3}\left[B(g_3)\right] = 0 \ .$$

Due to how the verifier chooses f, g_1, g_2 and g_3, we know that (f,g_1), (f,g_2) and (f,g_3) are all independent pairs of functions. Because of folding we have that

$$\mathrm{E}\left[A(f)B(g_1)\right] = \mathrm{E}\left[A(f)B(g_2)\right] = \mathrm{E}\left[A(f)B(g_3)\right] = 0 \ .$$

The triples (f,g_1,g_2) and $(-f,g_1,-g_2)$ are equally likely to be chosen by the verifier. This is also true for (f,g_1,g_3) and $(-f,g_1,-g_3)$ and thus

$$\mathrm{E}_{g_1,g_2}\left[B(g_1)B(g_2)\right] = \mathrm{E}_{g_1,g_3}\left[B(g_1)B(g_3)\right] = 0 \ .$$

As (f,g_1,g_2,g_3) occurs with the same probability as $(f,-g_1,-g_2,-g_3)$ we can conclude that

$$\mathrm{E}\left[A(f)B(g_1)B(g_2)B(g_3)\right] = \mathrm{E}\left[B(g_1)B(g_2)B(g_3)\right] = 0 \ .$$

As (f,g_2,g_3) occurs with the same probability as $(-f,-g_2,-g_3)$ we also know that

$$\mathrm{E}\left[A(f)B(g_2)B(g_3)\right] = 0 \ .$$

Summing up, we have reduced the expression for the acceptance probability of the verifier to

$$\begin{aligned}c_\emptyset + c_{\{3,4\}}\mathrm{E}_{g_2,g_3}\left[B(g_2)B(g_3)\right] + c_{\{1,2,3\}}\mathrm{E}_{f,g_1,g_2}\left[A(f)B(g_1)B(g_2)\right] +& \\ c_{\{1,2,4\}}\mathrm{E}_{f,g_1,g_3}\left[A(f)B(g_1)B(g_3)\right] \ .&\end{aligned} \tag{6.5}$$

The condition of Theorem 6.5 implies that $c_{\{3,4\}} \leq 0$. In order to get an upper bound on the acceptance probability, we show that $\mathrm{E}\left[B(g_2)B(g_3)\right] \geq 0$. This is easily seen by considering f and g_1 as fixed functions. Then g_2 and g_3 are chosen independently from the same distribution and thus the expectation of their product cannot be negative as $\mathrm{E}_{g_2,g_3}\left[B(g_2)B(g_3)\right] = \mathrm{E}_{g_2}\left[B(g_2)\right]^2$. We combine the condition of Lemma 6.6 with (6.5) and apply the bound $c_{\{3,4\}}\mathrm{E}_{g_2,g_3}\left[B(g_2)B(g_3)\right] \leq 0$:

$$c_{\{1,2,3\}}\mathrm{E}_{f,g_1,g_2}\left[A(f)B(g_1)B(g_2)\right] + c_{\{1,2,4\}}\mathrm{E}_{f,g_1,g_3}\left[A(f)B(g_1)B(g_3)\right] \quad \geq \quad \delta \ .$$

However, (f, g_1, g_2) and (f, g_1, g_3) have the same distribution and thus we have

$$(c_{\{1,2,3\}} + c_{\{1,2,4\}})\mathrm{E}_{f,g_1,g_2}\left[A(f)B(g_1)B(g_2)\right] \quad \geq \quad \delta \ .$$

Using the following claim we can bound the value of $|c_{\{1,2,3\}} + c_{\{1,2,4\}}|$.

Claim 6.7. $|c_S| \leq 1/2$*, for* $S \neq \emptyset$*.*

Proof of Claim 6.7. By (6.4) we have that

$$\begin{aligned} |c_S| &= 2^{-4}\left|\sum_{(x_1,x_2,x_3,x_4)\in\{\pm 1\}^4} P(x_1,x_2,x_3,x_4)\prod_{i\in S} x_i\right| \\ &\leq 2^{-4}\sum_{\substack{(x_1,x_2,x_3,x_4): \\ \prod_{i\in S} x_i = 1}} \prod_{i\in S} x_i \\ &= 1/2 \ . \end{aligned}$$

□

We conclude that

$$|\mathrm{E}_{f,g_1,g_2}\left[A(f)B(g_1)B(g_2)\right]| \quad \geq \quad \delta \ .$$

If the above expected value is negative, we can make it positive with the same absolute value by negating all entries of the table A. We note that the verifier chooses (f, g_1, g_2) in exactly the same way as the parity test verifier in Figure 3.3. We can therefore use Lemma 3.8 and conclude that if the condition of Lemma 6.6 is true, then there is a strategy for the two provers in the two-prover game that convinces the verifier with probability at least $4\epsilon\delta^2$. This concludes the proof of Lemma 6.6. □

We have produced a PCP with almost perfect completeness and soundness $c_\emptyset + \delta$, for an arbitrary $\delta > 0$, and where the verifier uses P as acceptance condition. By Theorem 2.22 we conclude that P is approximation resistant. □

6.2 Non-Trivially Approximable Predicates

By applying Lemma 4.6 or Theorem 4.9 we can directly conclude that many predicates of arity four can be approximated better than picking a random assignment. In this section we present two other methods for showing some predicates to be non-trivially approximable. Both are based on applications of Lemma 4.8. The description of each method is accompanied by an example when it is applied on a predicate of arity four.

Before describing the methods we give an approximation algorithm for Max kConjSAT from [18]. The algorithm is based on the linear relaxation algorithm by Trevisan [37] and it works well for instances with an optimal solution that has a significantly larger value than half of the total weight. This algorithm is used by the first method.

A Max kConjSAT Algorithm for $(1+\epsilon)/2$-Satisfiable Instances

Max kConjSAT is the problem of maximizing the weight of satisfied conjunctions. Each conjunction contains at most k binary variables, which can both occur positively or negated.

The linear program relaxation algorithm is identical to the one by Trevisan [37]. Each binary variable x_i is relaxed to t_i such that $0 \leq t_i \leq 1$. The constraint C_j has an associated variable z_j in the relaxation. If x_i occurs positively in C_j, then $z_j \leq t_i$ is a constraint in the relaxation, and if x_i occurs negated then $z_j \leq 1 - t_i$. It is easy to see that an assignment to Max kConjSAT can be transformed into a valid assignment for the linear program with the same objective value. This is done by letting $t_i = x_i$.

The difference of this algorithm compared with the algorithm of Trevisan is in how a solution to the linear program is transformed into an assignment of the associated Max kConjSAT instance. In [37] the value of x_i was decided by a flip of an unbiased coin with probability $(k-1)/k$ and only in the remaining case had the value of t_i some impact on the value of x_i. We instead make a threshold rounding by setting $x_i = 1$ if $t_i > 1/2$ and $x_i = 0$ otherwise. An analysis of this rounding scheme gives the following theorem.

Theorem 6.8. *A $(1+\epsilon)/2$-satisfiable instance of* Max kConjSAT*, where $\epsilon > 0$, can be approximated in polynomial time within $2\epsilon/(1+\epsilon)$ of the optimal value.*

Proof. Let w_{tot} be the sum of all the weights of the constraints, $\sum_{j=1}^{m} w_j$. Furthermore let ϵ' be defined such that $w_{\text{tot}}(1+\epsilon')/2$ is the value of an optimal solution to the Max kConjSAT instance. The value of an optimal solution, $\{z_j^*\}$, to the linear program, $\sum_{j=1}^{m} w_j z_j^*$, will be at least as large as $w_{\text{tot}}(1+\epsilon')/2$ because it is a relaxation of the original problem. Let $A = \{j : z_j^* \leq 1/2\}$ and $B = \{j : z_j^* > 1/2\}$. Furthermore we know that $z_j \leq 1$. We provide an upper bound of the optimal value.

$$\begin{aligned}
\frac{w_{\text{tot}}(1+\epsilon')}{2} &\leq \sum_{j=1}^{m} w_j z_j^* \\
&= \sum_{j\in A} w_j z_j^* + \sum_{j\in B} w_j z_j^* \\
&\leq \frac{1}{2}\sum_{j\in A} w_j + \sum_{j\in B} w_j \\
&= \frac{1}{2}\sum_{j=1}^{m} w_j + \frac{1}{2}\sum_{j\in B} w_j
\end{aligned}$$

Subtracting $w_{\text{tot}}/2$ from the left and the right equation and multiplying with two we get $\sum_{j\in B} w_j \geq \epsilon' w_{\text{tot}}$. The threshold rounding satisfy all the constraints that have $z_j^* > 1/2$, thus the value of the solution from the threshold rounding is at least $\sum_{j\in B} w_j$. The approximation ratio is thus at least

$$\frac{\sum_{j\in B} w_j}{w_{\text{tot}}(1+\epsilon')/2} \geq \frac{2\epsilon' w_{\text{tot}}}{w_{\text{tot}}(1+\epsilon')} = \frac{2\epsilon'}{1+\epsilon'} \geq \frac{2\epsilon}{1+\epsilon}$$

where we in the last inequality used that $\epsilon' \geq \epsilon > 0$. □

Method 1

The first step of this method is to show that all almost satisfiable instances of Max CSP(P), that do not have optimal solutions with a large weight on a single input, can be approximated non-trivially. After showing this, we use the Max kConjSAT algorithm in order to approximate the remaining instances.

We study the linear and bi-linear terms of the predicate P separately. This is accomplished by in Lemma 4.8 setting $C = \infty$ (linear) and $C = 0$ (bi-linear). Formally, we set C to a value large enough in the linear case. For some P, we can then conclude that there is an input y for which an optimal solution must have at least a $(1+\varepsilon)/2$ fraction of the total weight on, for some $\varepsilon > 0$. Otherwise, either Algorithm Lin or Algorithm BiLin approximates the instance non-trivially.

We now consider the Max CSP(P) instance as a Max kConjSAT instance, where each constraint only accepts the input y. There is a solution that satisfies at least $(1+\varepsilon)/2$ of the total weight, thus by applying Theorem 6.8 we conclude that we can find a solution that satisfies at least ε of the total weight. If ε is larger than the random assignment threshold for P, then P cannot be approximation resistant.

Example

In Table 6.2 the accepting inputs of a predicate P are listed along with the value of the linear and bi-linear terms for each input. By only considering the linear

Table 6.2: Linear and bi-linear terms for accepting inputs to the predicate P used by Method 1.

x_1	x_2	x_3	x_4	$2^4 \cdot P^{(1)}$	$2^4 \cdot P^{(2)}$
1	-1	-1	1	0	6
1	-1	-1	-1	2	4
-1	1	1	-1	0	6
-1	1	-1	1	0	-2
-1	-1	1	1	4	-2
-1	-1	1	-1	6	4
-1	-1	-1	-1	8	-2

terms we conclude that in order for an instance to be hard to be non-trivially approximated, an optimal solution has to have almost all weight on the inputs $(1, -1, -1, 1)$, $(-1, 1, 1, -1)$ and $(-1, 1, -1, 1)$. By looking at the bi-linear terms, we see that these inputs have value 6, 6 and -2 respectively. If the instance should not be non-trivially approximable, then the sum of the bi-linear terms should not be (more than negligibly) positive. Thus we conclude that at least $3/4 - \delta$ of the total weight is on input $(-1, 1, -1, 1)$, for any constant $\delta > 0$.

By Theorem 6.8, with $\epsilon = 1/2 - 2\delta$, we can find a solution with weight $1/2 - 2\delta$ of the total weight. By choosing δ small enough, for example $\delta = 1/33$, this is larger than the random assignment threshold for P which is $7/16$. Thus, P is non-trivially approximable and cannot be approximation resistant.

Method 2

This method is similar to the method we used in order to prove Theorem 4.9. We start by applying Lemma 4.8 with respect to the predicate P. For some predicates P, the lemma guarantees that almost satisfiable instances of Max CSP(P), that cannot be approximated non-trivially by Algorithms Lin or BiLin, have an optimal solution with almost all weight on some special inputs $y_1, \ldots y_l$. We produce a maximal gadget to 2SAT such that the inputs $y_1, \ldots y_l$ satisfy all the 2SAT constraints of the gadget. We then run Zwick's algorithm for almost satisfying Max 2SAT instances [43]. If the original instance is almost satisfiable, then the gadget ensures that the Max 2SAT instance is almost satisfiable as well. Zwick's algorithm returns a solution that satisfies $1 - O(\varepsilon^{1/3})$ of the total weight if an optimal solution satisfies $1 - \varepsilon$ of the total weight. If $y_1, \ldots y_l$ are the only inputs that satisfy all the 2SAT constraints of the gadget, then the solution from Zwick's algorithm is an almost satisfying solution for the original instance. However, if there exist other inputs that also satisfy the constraint of the gadget, then we need to consider the possibility that the solution have significant weight on those inputs. If the inputs in question are accepted by P, then there is no problem. If not, it may be a good idea to consider the negation of the solution.

Table 6.3: Linear and bi-linear terms for accepting inputs to the predicate P used by Method 2.

x_1	x_2	x_3	x_4	$2^4 \cdot P^{(1)}$	$2^4 \cdot P^{(2)}$	$2^4 \cdot (2P^{(1)} + P^{(2)})$
1	−1	1	−1	−2	4	0
1	−1	−1	1	−2	4	0
−1	1	1	−1	2	4	8
−1	1	−1	1	2	4	8
−1	1	−1	−1	4	6	14
−1	−1	1	1	2	−4	0
−1	−1	−1	−1	6	−4	8

Example

The accepting inputs to P are listed in Table 6.3. Using Lemma 4.8 with $C = 2$ we can conclude that an almost satisfying solution has to have almost all weight on inputs $(1, -1, 1, -1)$, $(1, -1, -1, 1)$ and $(-1, -1, 1, 1)$ or either Algorithm Lin or Algorithm BiLin approximates the instance non-trivially. Thus, we can assume that the total weight on these inputs for an optimal solution is a $1 - \delta$ fraction of the total weight, for any constant $\delta > 0$. These three inputs satisfy all constraints of the following 2SAT gadget:

$$x_2 \quad \overline{x}_1 \vee \overline{x}_3 \quad \overline{x}_1 \vee \overline{x}_4 \quad \overline{x}_3 \vee \overline{x}_4$$

However, the input $(1, -1, 1, 1)$ satisfies the gadget as well. The predicate P does not accept this input so this could be a potential problem. Using Zwick's 2SAT algorithm for almost satisfying instances, Theorem 2.24, we get a solution that has at least a $1-5\delta^{1/3}$ fraction of the total weight on inputs $(1, -1, 1, -1)$, $(1, -1, -1, 1)$, $(-1, -1, 1, 1)$ and $(1, -1, 1, 1)$. We need to find a solution that satisfies at least a $7/16 + \varepsilon$ fraction of the total weight, for some small constant $\varepsilon > 0$. If the obtained solution does not do this it is because at least a $1 - 5\delta^{1/3} - (7/16 + \varepsilon)$ fraction of the weight is on the non-accepting input $(1, -1, 1, 1)$. In that case we negate the solution and then get more than a $9/16 - 5\delta^{1/3} - \varepsilon$ fraction of the weight on input $(-1, 1, -1, -1)$. By choosing δ and ε small enough we can ensure that $9/16 - 5\delta^{1/3} - \varepsilon > 7/16 + \varepsilon$. As $(-1, 1, -1, -1)$ is an accepting input we have obtained an approximation ratio larger than the random assignment threshold.

6.3 Tables of All Predicate Types of Arity Four

In this section we present our classification of predicates into approximation resistant and non-trivially approximable predicates. All predicates with between six and fourteen accepting inputs are represented in Tables 6.4 to 6.12. Predicates with fewer accepting inputs are all non-trivially approximable by Theorem 4.4.

There is only one predicate type with fifteen accepting inputs. The corresponding CSP is equivalent with the Max E4SAT problem which has been shown to be approximation resistant [22].

Notation Used in the Tables

A predicate is expressed as a bitstring of length 16 corresponding to the truth table of the predicate. Let P be represented by $s_0 s_1 \dots s_{15} \in \{0,1\}^{16}$. Then P is defined as

$$\begin{array}{rcl} P(+1,+1,+1,+1) & = & s_0 \\ P(+1,+1,+1,-1) & = & s_1 \\ P(+1,+1,-1,+1) & = & s_2 \\ & \dots & \\ P(-1,-1,-1,-1) & = & s_{15} \ . \end{array}$$

A predicate type is represented by its predicate with the lexicographically smallest string representation. The predicate types are listed in lexicographical order. To the right of the predicate type either 'A', 'R' or '?' appears. 'A' means that we have shown the predicate type to be non-trivially approximable and 'R' means that it is approximation resistant. A '?' indicates that we have not been able to classify the predicate type. In the next column there is a brief explanation of how the classification was made. Either a theorem, lemma or a reference to a method appears. Sometimes a value to C is specified, and this relates to Theorem 4.9 or Method 2.

For many predicate types there are more than one way to show its classification. If this is the case for a predicate, we apply the following priority order to decide which method we account for (highest priority first):

- Showing approximation resistance:
  Theorem 6.1, Theorem 6.2, Theorem 6.3, Theorem 6.4 and Theorem 6.5.
- Showing non-trivial approximation algorithms:
  Lemma 4.6, Theorem 4.9, Method 1 and Method 2.

6.4 A Final Remark

In order to reduce the number of not classified predicates of arity four, we believe that using Karloff and Zwick's method of obtaining a semidefinite relaxation [28] could be rewarding. It should also be possible to combine this method with the method in this paper that produce conditions which optimal solutions must adhere to.

Table 6.4: Approximability of predicates with exactly 6 accepting inputs.

0000000000111111	A	Lemma 4.6	0000001101101100	A	Thm 4.9, $C = 1.0$
0000000001101111	A	Lemma 4.6	0000001111000011	A	Lemma 4.6
0000000001111110	A	Lemma 4.6	0000001111000101	A	Lemma 4.6
0000000100011111	A	Lemma 4.6	0000001111000110	A	Lemma 4.6
0000000100101111	A	Lemma 4.6	0000001111010100	A	Lemma 4.6
0000000100111101	A	Lemma 4.6	0000001111011000	A	Lemma 4.6
0000000100111110	A	Thm 4.9, $C = 1.0$	0000011001100011	A	Lemma 4.6
0000000101101011	A	Thm 4.9, $C = 1.0$	0000011001100110	A	Lemma 4.6
0000000101101110	A	Thm 4.9, $C = 1.0$	0000011001101001	A	Thm 4.9, $C = 1.0$
0000000110001111	A	Lemma 4.6	0000011001110010	A	Lemma 4.6
0000000110010111	A	Lemma 4.6	0000011001111000	A	Thm 4.9, $C = 1.0$
0000000110011011	A	Lemma 4.6	0000011010010011	A	Thm 4.9, $C = 1.0$
0000000110011110	A	Thm 4.9, $C = 1.0$	0000011010010110	A	Thm 4.9, $C = 1.0$
0000000110101011	A	Lemma 4.6	0000011010110001	A	Thm 4.9, $C = 1.0$
0000000110101101	A	Lemma 4.6	0000011010110010	A	Lemma 4.6
0000000110101110	A	Lemma 4.6	0000011010110100	A	Thm 4.9, $C = 1.0$
0000000110111100	A	Lemma 4.6	0000011011110000	A	Lemma 4.6
0000000111101001	A	Lemma 4.6	0000011110110000	A	Lemma 4.6
0000000111101010	A	Lemma 4.6	0000011111100000	A	Lemma 4.6
0000001100111100	A	Lemma 4.6	0001011001101000	A	Lemma 4.6
0000001101010110	A	Lemma 4.6	0001011010000011	A	Thm 4.9, $C = 0.5$
0000001101011001	A	Thm 4.9, $C = 1.0$	0001011010000110	A	Thm 4.9, $C = 1.0$
0000001101011010	A	Lemma 4.6	0001011010001001	A	Thm 4.9, $C = 1.5$
0000001101101001	A	Thm 4.9, $C = 1.0$	0001011010011000	A	Thm 4.9, $C = 1.0$
0000001101101010	A	Lemma 4.6	0001011110000001	A	Lemma 4.6

Table 6.5: Approximability of predicates with exactly 7 accepting inputs.

0000000001111111	A	Lemma 4.6	0000011001111010	A	Lemma 4.6
0000000100111111	A	Lemma 4.6	0000011010010111	?	
0000000101101111	A	Lemma 4.6	0000011010110011	A	Lemma 4.6
0000000101111110	A	Method 1	0000011010110101	A	Thm 4.9, $C = 0.0$
0000000110011111	A	Lemma 4.6	0000011010110110	A	Lemma 4.6
0000000110101111	A	Lemma 4.6	0000011010111001	?	
0000000110111101	A	Lemma 4.6	0000011011110001	A	Lemma 4.6
0000000110111110	A	Lemma 4.6	0000011011110010	A	Lemma 4.6
0000000111101011	A	Lemma 4.6	0000011101111000	A	Method 1
0000000111101110	A	Lemma 4.6	0000011110110001	A	Lemma 4.6
0000001100111101	A	Lemma 4.6	0000011110110100	A	Lemma 4.6
0000001101010111	A	Lemma 4.6	0000011111100001	A	Lemma 4.6
0000001101011011	A	Lemma 4.6	0000011111100010	A	Lemma 4.6
0000001101011110	A	Lemma 4.6	0000011111110000	A	Lemma 4.6
0000001101101011	A	Lemma 4.6	0001011001101001	A	Thm 4.9, $C = 1.5$
0000001101101101	A	Method 1	0001011001101010	A	Lemma 4.6
0000001101101110	A	Lemma 4.6	0001011010000111	A	Method 2, $C = 0.0$
0000001101111100	A	Lemma 4.6	0001011010001011	?	
0000001111000111	A	Lemma 4.6	0001011010001110	A	Lemma 4.6
0000001111010101	A	Lemma 4.6	0001011010010110	?	
0000001111010110	A	Lemma 4.6	0001011010011001	?	
0000001111011001	A	Lemma 4.6	0001011010011010	A	Thm 4.9, $C = 0.0$
0000001111011100	A	Lemma 4.6	0001011010101001	A	Method 2, $C = 2.0$
0000011001100111	A	Lemma 4.6	0001011010101100	?	
0000011001101011	A	Lemma 4.6	0001011110000011	A	Lemma 4.6
0000011001110011	A	Lemma 4.6	0001011110001001	A	Lemma 4.6
0000011001110110	A	Lemma 4.6	0001011110011000	A	Lemma 4.6
0000011001111001	A	Method 2, $C = 2.0$	0001100111100001	A	Thm 4.9, $C = 0.5$

Table 6.6: Approximability of predicates with exactly 8 accepting inputs.

0000000011111111	A	Lemma 4.6	0000011111110001	A	Lemma 4.6
0000000101111111	A	Thm 4.9, $C = 1.0$	0000011111110010	A	Lemma 4.6
0000000110111111	A	Lemma 4.6	0000011111111000	A	Thm 4.9, $C = 1.0$
0000000111101111	A	Lemma 4.6	0000111111110000	A	Lemma 4.6
0000000111111110	A	Thm 4.9, $C = 1.0$	0001011001101011	R	Thm 6.3
0000001100111111	A	Lemma 4.6	0001011001101110	A	Thm 4.9, $C = 1.0$
0000001101011111	A	Lemma 4.6	0001011010001111	?	
0000001101101111	A	Thm 4.9, $C = 1.0$	0001011010010111	R	Thm 6.4
0000001101111101	A	Thm 4.9, $C = 1.0$	0001011010011011	R	Thm 6.4
0000001101111110	A	Thm 4.9, $C = 1.0$	0001011010011110	R	Thm 6.4
0000001111001111	A	Lemma 4.6	0001011010101011	?	
0000001111010111	A	Lemma 4.6	0001011010101101	R	Thm 6.3
0000001111011011	A	Lemma 4.6	0001011010101110	A	Thm 4.9, $C = 1.0$
0000001111011101	A	Lemma 4.6	0001011010111100	?	
0000001111011110	A	Thm 4.9, $C = 1.0$	0001011011101001	R	Thm 6.3
0000001111111100	A	Lemma 4.6	0001011011101010	A	Thm 4.9, $C = 1.0$
0000011001101111	A	Lemma 4.6	0001011110000111	A	Thm 4.9, $C = 1.0$
0000011001110111	A	Lemma 4.6	0001011110001011	A	Thm 4.9, $C = 1.0$
0000011001111011	A	Thm 4.9, $C = 1.0$	0001011110001110	A	Lemma 4.6
0000011001111110	A	Lemma 4.6	0001011110010110	R	Thm 6.3
0000011010011111	R	Thm 6.4	0001011110011001	A	Thm 4.9, $C = 1.0$
0000011010110111	A	Thm 4.9, $C = 1.0$	0001011110011010	A	Thm 4.9, $C = 1.0$
0000011010111011	A	Thm 4.9, $C = 1.0$	0001011110101001	A	Thm 4.9, $C = 1.0$
0000011010111101	?		0001011110101100	A	Thm 4.9, $C = 1.0$
0000011011110011	A	Thm 4.9, $C = 1.0$	0001011111101000	A	Lemma 4.6
0000011011110110	A	Lemma 4.6	0001100011100111	R	Thm 6.4
0000011011111001	R	Thm 6.3	0001100111100011	R	Thm 6.4
0000011101110110	A	Lemma 4.6	0001100111100110	R	Thm 6.3
0000011101111001	?		0001100111101001	?	
0000011101111010	A	Thm 4.9, $C = 1.0$	0001100111101010	A	Thm 4.9, $C = 1.0$
0000011110110011	A	Lemma 4.6	0001100111110001	A	Thm 4.9, $C = 1.0$
0000011110110101	A	Thm 4.9, $C = 1.0$	0001100111111000	A	Lemma 4.6
0000011110110110	A	Thm 4.9, $C = 1.0$	0001101111011000	A	Lemma 4.6
0000011110111100	A	Thm 4.9, $C = 1.0$	0001101111100100	R	Thm 6.4
0000011111100011	A	Lemma 4.6	0001111011100001	R	Thm 6.3
0000011111100110	A	Lemma 4.6	0011110011000011	R	Thm 6.2
0000011111101001	A	Thm 4.9, $C = 1.0$	0110100110010110	R	Thm 6.1

Table 6.7: Approximability of predicates with exactly 9 accepting inputs.

0000000111111111	A	Lemma 4.6
0000001101111111	A	Lemma 4.6
0000001111011111	A	Lemma 4.6
0000001111111101	A	Lemma 4.6
0000011001111111	A	Lemma 4.6
0000011010111111	?	
0000011011110111	A	Lemma 4.6
0000011011111011	?	
0000011101110111	A	Lemma 4.6
0000011101111011	?	
0000011101111110	A	Lemma 4.6
0000011110110111	A	Lemma 4.6
0000011110111101	?	
0000011111100111	A	Lemma 4.6
0000011111101011	A	Lemma 4.6
0000011111110011	A	Lemma 4.6
0000011111110110	A	Lemma 4.6
0000011111111001	?	
0000011111111010	A	Lemma 4.6
0000111111110001	A	Lemma 4.6
0001011001101111	?	
0001011001111110	A	Lemma 4.6
0001011010011111	R	Thm 6.5
0001011010101111	?	
0001011010111101	?	
0001011010111110	?	
0001011011101011	?	
0001011011101110	A	Thm 4.9, $C = 0.0$
0001011110001111	A	Lemma 4.6
0001011110010111	A	Thm 4.9, $C = 0.6$
0001011110011011	?	
0001011110011110	?	
0001011110101011	A	Lemma 4.6
0001011110101101	?	
0001011110101110	A	Lemma 4.6
0001011110111100	?	
0001011111101001	A	Thm 4.9, $C = 1.5$
0001011111101010	A	Lemma 4.6
0001100011101111	?	
0001100111100111	R	Thm 6.5
0001100111101011	?	
0001100111101110	?	
0001100111110011	A	Thm 4.9, $C = 0.5$
0001100111110110	?	
0001100111111001	A	Lemma 4.6
0001100111111010	A	Lemma 4.6
0001101111010110	?	
0001101111011001	A	Lemma 4.6
0001101111100101	R	Thm 6.5
0001101111101100	?	
0001111011100011	R	Thm 6.5
0001111011100110	?	
0001111011101001	?	
0001111011110001	?	
0011110011000111	R	Thm 6.2
0110100110010111	R	Thm 6.1

Table 6.8: Approximability of predicates with exactly 10 accepting inputs.

0000001111111111	A	Lemma 4.6	0001100111111110	R	Thm 6.3
0000011011111111	A	Thm 4.9, $C = 1.0$	0001101111010111	?	
0000011101111111	A	Thm 4.9, $C = 1.0$	0001101111011011	A	Lemma 4.6
0000011110111111	A	Thm 4.9, $C = 1.0$	0001101111011110	A	Thm 4.9, $C = 1.0$
0000011111101111	A	Lemma 4.6	0001101111100111	R	Thm 6.4
0000011111110111	A	Lemma 4.6	0001101111101101	R	Thm 6.3
0000011111111011	A	Thm 4.9, $C = 1.0$	0001101111101110	R	Thm 6.4
0000011111111110	A	Thm 4.9, $C = 1.0$	0001101111111100	A	Thm 4.9, $C = 1.0$
0000111111110011	A	Lemma 4.6	0001111011100111	R	Thm 6.3
0000111111110110	A	Thm 4.9, $C = 1.0$	0001111011101011	R	Thm 6.3
0001011001111111	A	Thm 4.9, $C = 0.8$	0001111011101110	A	Thm 4.9, $C = 0.5$
0001011010111111	R	Thm 6.4	0001111011110011	R	Thm 6.3
0001011011101111	R	Thm 6.3	0001111011110110	R	Thm 6.4
0001011011111110	A	Thm 4.9, $C = 0.5$	0001111011111001	R	Thm 6.3
0001011101111110	A	Lemma 4.6	0001111011111010	A	Thm 4.9, $C = 1.0$
0001011110011111	R	Thm 6.4	0001111111110001	A	Thm 4.9, $C = 1.0$
0001011110101111	A	Thm 4.9, $C = 1.0$	0001111111110010	A	Thm 4.9, $C = 1.0$
0001011110111101	?		0001111111111000	A	Lemma 4.6
0001011110111110	A	Thm 4.9, $C = 1.5$	0011110011001111	R	Thm 6.2
0001011111101011	R	Thm 6.3	0011110011010111	R	Thm 6.2
0001011111101110	A	Thm 4.9, $C = 1.0$	0011110011011011	R	Thm 6.2
0001100011111111	A	Thm 4.9, $C = 1.0$	0011110111010110	R	Thm 6.3
0001100111101111	R	Thm 6.4	0011110111011010	R	Thm 6.3
0001100111110111	R	Thm 6.3	0110100110011111	R	Thm 6.1
0001100111111011	A	Thm 4.9, $C = 1.0$	0110101111010110	R	Thm 6.1

Table 6.9: Approximability of predicates with exactly 11 accepting inputs.

0000011111111111	A	Lemma 4.6	0001111011111110	A	Thm 4.9, $C = 0.6$
0000111111110111	A	Lemma 4.6	0001111111110011	?	
0001011011111111	?		0001111111110110	?	
0001011101111111	A	Lemma 4.6	0001111111111001	A	Thm 4.9, $C = 1.5$
0001011110111111	?		0001111111111010	A	Lemma 4.6
0001011111101111	?		0011110011011111	R	Thm 6.2
0001011111111110	A	Lemma 4.6	0011110111010111	R	Thm 6.2
0001100111111111	A	Method 2, $C = 2/3$	0011110111011011	R	Thm 6.2
0001101111011111	A	Lemma 4.6	0011110111011110	R	Thm 6.5
0001101111101111	R	Thm 6.5	0011110111101101	R	Thm 6.5
0001101111111101	?		0110100110111111	R	Thm 6.1
0001111011101111	R	Thm 6.5	0110101110111101	R	Thm 6.5
0001111011110111	R	Thm 6.5	0110101111010111	R	Thm 6.1
0001111011111011	?				

Table 6.10: Approximability of predicates with exactly 12 accepting inputs.

0000111111111111	A	Lemma 4.6	0011110111111101	R	Thm 6.4
0001011111111111	A	Thm 4.9, $C = 0.8$	0011110111111110	R	Thm 6.3
0001101111111111	R	Thm 6.4	0011111111111100	A	Lemma 4.6
0001111011111111	R	Thm 6.3	0110100111111111	R	Thm 6.1
0001111111110111	R	Thm 6.4	0110101110111111	R	Thm 6.1
0001111111111011	R	Thm 6.3	0110101111011111	R	Thm 6.1
0001111111111110	?		0110101111111101	R	Thm 6.2
0011110011111111	R	Thm 6.2	0110111111110110	R	Thm 6.1
0011110111011111	R	Thm 6.2	0110111111111001	R	Thm 6.2
0011110111101111	R	Thm 6.2			

Table 6.11: Approximability of predicates with exactly 13 accepting inputs.

0001111111111111	?		0110101111111111	R	Thm 6.1
0011110111111111	R	Thm 6.2	0110111111110111	R	Thm 6.1
0011111111111101	A	Thm 4.9, $C = 1.5$	0110111111111011	R	Thm 6.2

Table 6.12: Approximability of predicates with exactly 14 accepting inputs.

0011111111111111	R	Thm 6.2	0111111011111111	R	Thm 6.2
0110111111111111	R	Thm 6.1	0111111111111110	R	Thm 6.1

Chapter 7

Structure of Approximation Resistance

In the previous chapters we have seen some tendencies concerning which predicates that are approximation resistant. The most striking one is that predicates with many accepting inputs tend to be approximation resistant, while predicates with few accepting inputs tend to be non-trivially approximable. In this chapter, we take a closer look at this observation. A natural conjecture is that if a predicate is implied by an approximation resistant predicate, then it is also approximation resistant. We show that this conjecture is false. We also study the approximability of predicates close to each other and how the approximability of a predicate changes if we make small changes to it.

7.1 Approximation Resistance is Non-Monotone

For all approximation resistant predicates P of arity three, it is true that predicates implied by P is approximation resistant as well. This is true for all Samorodnitsky-Trevisan predicates as well. A natural guess could be that it is true in general. However, this is not the case.

A predicate shown to be approximation resistant in [15] is the GLST predicate, defined in (5.5). Zwick [42] showed that the not all equal predicate of arity three, $\mathrm{3NAE}(x_1, x_2, x_3) = (x_1 \oplus x_2) \vee (x_1 \oplus x_3)$, is not approximation resistant. But $\mathrm{GLST}(x_1, x_2, x_3, x_4)$ implies $\mathrm{3NAE}(\overline{x}_2, x_3, x_4)$ which gives the following theorem that falsifies the guess.

Theorem 7.1. *Not all approximation resistant predicates are also hereditary approximation resistant.*

7.2 Neighborhood Approximability

In this section we study how the approximability of Max CSP(P) and Max CSP(P') are related if P and P' are close predicates. In order to make the presentation clearer we make the following definitions.

Definition 7.2. By $\alpha_A(P)$, we denote the approximation ratio of algorithm A on Max CSP(P).

Definition 7.3. Let P be a predicate $\{\pm 1\}^k \to \{0,1\}$. By $\alpha(P)$, we denote the maximal approximation ratio of any polynomial time algorithm on Max CSP(P),

$$\alpha(P) \quad = \quad \max_{A \text{ polytime}} \alpha_A(P) \ ,$$

and for a set of predicates $\mathcal{P}$ let

$$\alpha^{min}(\mathcal{P}) \quad = \quad \min_{P \in \mathcal{P}} \alpha(P)$$

and

$$\alpha^{max}(\mathcal{P}) \quad = \quad \max_{P \in \mathcal{P}} \alpha(P) \ .$$

A natural way to measure how much two predicates P and P' differ from each other is to consider the fraction of inputs that make P and P' give different output,

$$\Pr_x \left[P(x) \neq P'(x) \right] \ . \tag{7.1}$$

The smallest change we can do to a predicate is to change the output for one value. The probability in (7.1) then equals 2^{-k}, where k is the arity of the predicate P. In order to enable smaller changes we let P_t be predicate P but with arity $t \geq k$. The additional $t-k$ input bits are simply ignored by P_t. We see that $\alpha(P) = \alpha(P_t)$. In some sense we can make arbitrarily small changes to P by enlarging its arity and instead considering P_t for some large value of t. Below, we define the neighborhood of a predicate P by considering predicates close to P_t. We make this definition because we want to study how $\alpha(P)$ varies for predicates P close to each other.

Definition 7.4. Let P be a predicate $\{\pm 1\}^k \to \{0,1\}$ and let $P_t(b_1, \ldots, b_t) = P(b_1, \ldots, b_k)$ for $t \geq k$. The *neighborhood* $N_t(P,q)$ contains the predicates that can be created by changing the output of P_t on at most q inputs. The *upper neighborhood* $N_t^+(P,q)$ contains the predicates that can be created by adding at most q accepting inputs to P_t. The *lower neighborhood* $N_t^-(P,q)$ contains the predicates that can be created by removing at most q accepting inputs from P_t.

We show that some predicates in the upper neighborhood of any predicate, provided that the neighborhood is large enough, are approximation resistant.

Theorem 7.5. *Assume* P $\neq$ NP, *then for any predicate* $P : \{\pm 1\}^k \to \{0,1\}$

$$\lim_{t\to\infty} \alpha^{min}(N_t^+(P, 2^{4\lfloor\sqrt{t+1}\rfloor-2})) \quad = \quad \alpha_R(P) \ ,$$

where R *is the random assignment algorithm.*

The idea behind the proof is to take a Samorodnitsky-Trevisan predicate P_{ST} and consider the predicate $P_t \vee P_{ST}$. This predicate is in the upper neighborhood of P_t, as P_{ST} has a small fraction of accepting inputs. Furthermore, it is also approximation resistant due to Theorem 5.2.

Proof. Choose s as large as possible such that $2s + s^2 \leq t$. Thus, $s = \lfloor\sqrt{t+1}\rfloor - 1$. We create a predicate P' that is implied by $P_{ST^{s,s}}$ acting on the first $2s + s^2$ bits. It is created by adding accepting inputs to P_t. The number of accepting inputs of $P_{ST^{s,s}}$ is 2^{2s}. As the last $t - (2s + s^2)$ bits in an input do not effect $P_{ST^{s,s}}$, we have to add $2^{t-(2s+s^2)}$ inputs to P for each accepting input of $P_{ST^{s,s}}$. We know that $t < 2(s+1) + (s+1)^2 = 2s + 2 + s^2 + 2s + 1$ and thus $t - (2s + s^2) \leq 2s + 2$. The number of accepting inputs that need to be added in order to create P' from P_t is at most $2^{2s} \cdot 2^{2s+2} = 2^{4s+2} = 2^{4\lfloor\sqrt{t+1}\rfloor-2}$.

As P' is implied by $P_{ST^{s,s}}$ and is contained in $N_t^+(P, 2^{4\lfloor\sqrt{t+1}\rfloor-2})$ we know by Theorem 5.2 that it is NP-hard to approximate Max CSP(P') within $\alpha_R(P') + \epsilon$ for any constant $\epsilon > 0$. The theorem follows by observing that $\lim_{t\to\infty} \alpha_R(P') = \alpha_R(P)$. $\square$

The predicates in the neighborhood used in the theorem may differ on $2^{O(\sqrt{t})}$ inputs. Let us for a moment consider smaller sizes of the upper neighborhood for the predicates 2AND and 2OR.

For a predicate $P \in N_t^+(\text{2AND}, 1)$ we have that $\alpha_R(P) \leq 0.25 + 2^{-t}$, where R is the random assignment algorithm. The following theorem lets us conclude that predicates in the upper neighborhood $N_t^+(\text{2AND}, 1)$, for t large enough, are not approximation resistant.

Theorem 7.6. *For* $t > 2$,

$$\alpha^{min}(N_t^+(\text{2AND}, 1)) \quad \geq \quad 0.274 \ .$$

Proof. Let $P \in N_t^+(\text{2AND}, 1)$ and I be an instance of Max CSP(P) such that the value of an optimal solution is ρw_{tot}, where w_{tot} is the sum of all weights in I. P can be expressed as $\text{2AND}_t \vee t\text{Conj}$, where 2AND_t takes and of the two first input bits and ignores the rest, and tConj is a conjunction on t literals. If a constraint in I is satisfied, it is either satisfied because of 2AND_t or tConj. We know that an optimal solution to I satisfies constraints of weight ρw_{tot}. Let $\rho_{\text{2AND}} w_{\text{tot}}$ and $\rho_{t\text{Conj}} w_{\text{tot}}$ be the total weight of the constraints that were satisfied due to the respective predicates. We know that $\rho = \rho_{\text{2AND}} + \rho_{t\text{Conj}}$ (we assume that 2AND is not implied by tConj). We approximate I as a Max CSP$(t\text{Conj})$ instance, i.e., a Max tConjSAT instance, using the linear relaxation algorithm defined in Section 6.2. We also approximate I as a Max CSP(2AND_t) instance using the 0.874-approximation algorithm for Max CSP(2AND) in [31]. In addition, a random assignment algorithm is also performed on I. The best solution of the three algorithms is used. By Theorem 6.8 we know that if $\rho_{t\text{Conj}} = (1+\epsilon)/2$, then the combined algorithm returns a solution of weight at least ϵw_{tot} due to the linear relaxation algorithm. Furthermore, the Max CSP(2AND) algorithm produces a solution of weight at least $0.874\rho_{2\text{AND}} w_{\text{tot}}$ and a random assignment satisfies weight of $1/4 + 2^{-t}$. Thus, the returned solution will have at least weight $\max(2\rho_{t\text{Conj}} - 1, 0.874\rho_{2\text{AND}}, 1/4+2^{-t})w_{\text{tot}}$. The approximation ratio is minimized if $\rho = 0.911$, $\rho_{2\text{AND}} = 0.286$ and $\rho_{t\text{Conj}} = 0.625$. In this case the approximation ratio is at least 0.274. □

Theorem 7.7. *For $t > 2$,*

$$\alpha^{max}(N_t^+(2\text{OR}, 1) \setminus \{2\text{OR}_t\}) \quad \leq \quad \alpha(3\text{OR}) \ .$$

Proof. Let $P \in N_t^+(2\text{OR}, 1)$ and we assume that $P \neq 2\text{OR}_t$. Given an instance I of Max CSP(3OR), we show that we can produce an instance I' of Max CSP(P), such that given a solution to I' with a certain weight it is easy to produce a solution to I of at least the same weight, and vice versa.

Given a 3OR-constraint tuple (z_1, z_2, z_3) in I, we put the following P-constraint tuple in I', $(z_1, z_2, bz_3, y_1, \ldots y_{t-3})$. The variables $y_1, \ldots y_{t-3}$ are auxiliary variables that are identical in all constraints in I'. The value of the negation mask b depends on P, if P accepts no input that starts with $(1,1,1)$, then $b = 1$ and otherwise $b = -1$. A solution to I' will now also be a solution to I which will satisfy all constraints in I corresponding to the ones satisfied in I'. Thus, it cannot be easier to approximate Max CSP(P) than Max CSP(3OR). □

We have the following theorems for the approximability of Max CSP(2OR) and Max CSP(3OR).

Theorem 7.8 (Lewin et al. [31]).

$$\alpha(2\text{OR}) \quad \geq \quad 0.940$$

Theorem 7.9 (Håstad [22]). *Assume* P $\neq$ NP*, then for any constant $\epsilon > 0$*

$$\alpha(3\text{OR}) \quad \leq \quad 0.875 + \epsilon \ .$$

We combine these theorems with Theorem 7.7 and conclude that the achievable approximation ratio for Max CSP(2OR) is strictly larger than if we consider Max CSP(P), where P is 2OR_t with one added accepting input.

Theorem 7.10. *Assume* P $\neq$ NP*, then for $t > 2$*

$$\alpha^{max}(N_t^+(2\text{OR}, 1) \setminus \{2\text{OR}_t\}) \quad < \quad 0.06 + \alpha(2\text{OR}) \ .$$

From this we can conclude that the smallest possible change to a predicate can effect the approximability of the corresponding Max CSP. We express this by the following corollary.

Corollary 7.11. *For a predicate P, it is not generally true that*

$$\lim_{t\to\infty} \alpha^{max}(N_t^+(P,1) \setminus \{P_t\}) \quad = \quad \alpha(P) \ ,$$

unless P = NP.

Remark 7.12. Corollary 7.11 can also be shown by considering the unary predicate $\mathrm{UN}(x_1) = x_1$. We have that all predicates in $N_2^+(\mathrm{UN},1)$, except for UN_2 are of the same type as 2OR. As Max CSP(UN) can be solved exactly, we have that $\alpha(\mathrm{UN}) = 1$. Furthermore, it is known that $\alpha(\mathrm{2OR}) < 1$. By applying the same technique as in the proof of Theorem 7.7 it is possible to show that $\alpha^{max}(N_t^+(\mathrm{UN},1) \setminus \{\mathrm{UN}_2\}) \le \alpha(\mathrm{2OR})$, for all $t > 1$.

Turning to the lower neighborhoods we show the following theorem.

Theorem 7.13. *Let $P : \{\pm 1\}^k \to \{0,1\}$ be a predicate, $\delta(t) \in o(t)$ and $\gamma(t) \in o(2^t)$. Then*

$$\lim_{t\to\infty} \alpha^{min}(N_t^-(P, 2^{\delta(t)})) \ = \ \lim_{t\to\infty} \alpha^{max}(N_t(P,\gamma(t))) \ = \ \alpha(P) \ .$$

Proof. First we show that an approximation algorithm A for Max CSP(P) implies an algorithm A' for Max CSP(P'), where $P' \in N_t^-(P, 2^{\delta(t)})$, with the same approximation ratio when $t \to \infty$. After this we conclude the proof by showing that if Max CSP(P') can be approximated within β, for a $P' \in N_t(P,\gamma(t))$, then Max CSP(P) can be approximated within $\beta - o_t(1)$.

Let A be a β-approximation algorithm for Max CSP(P), where $P : \{\pm 1\}^k \to \{0,1\}$. Let $P' \in N_t^-(P, 2^{\delta(t)})$. Consider the following algorithm A' for Max CSP(P'): Truncate each constraint so it only contains the first k literals. Apply A on the resulting instance (of truncated constraints). The assignment returned by A is now perturbated by letting each variable change value with a small probability $\epsilon(t) = \max(\delta(t)/t, t^{-1/2}) \in o_t(1) \cap \omega(t^{-1})$. If any variables in the instance were eliminated by the truncation step, these are given random values.

The intuition of the algorithm is that the perturbation will most likely not effect any of the first k literals, when t is large enough, but will most likely effect the rest of the literals in such a way that every specific t-tuple has a very small probability of being chosen.

Claim 7.14. *Algorithm A' approximates* Max CSP(P') *within a factor of $\beta - o_t(1)$.*

Proof. Let I' be the original Max CSP(P') instance, and I be the instance of Max CSP(P) resulting from the truncation. The value of an assignment ϕ to I' is always smaller or equal to the value of ϕ to I. Thus, we have that $w_{\text{opt}}(I) \geq w_{\text{opt}}(I')$. The value of the solution returned by A is at least $\beta w_{\text{opt}}(I) \geq \beta w_{\text{opt}}(I')$.

Look at an arbitrary constraint in I that was satisfied by $A(I)$. When is the corresponding constraint in I' satisfied after the perturbation step? Given that the perturbation did not change value on any of the k first literals, the value of the constraint tuple should not be in $P_t^{-1}(1) \setminus P'^{-1}(1)$. The probability that not any of the k first literals were perturbated is $(1-\epsilon(t))^k$ which is in $1 - o_t(1)$ because k is a constant. We bound the probability that the value of the constraint tuple is in $P_t^{-1}(1) \setminus P'^{-1}(1)$ by calculating the probability for the most likely constraint tuple (no literals were affected by the perturbation) and multiplying this with $|P_t^{-1}(1) \setminus P'^{-1}(1)|$.

The probability that the perturbation did not change value on any of the literals is

$$(1-\epsilon(t))^t \;<\; e^{-t\epsilon(t)} \;=\; \min(e^{-\delta(t)}, e^{-\sqrt{t}}) \;.$$

The inequality is valid as $1 - \epsilon \leq e^{-\epsilon}$. By the condition of the theorem, the size of $P^{-1}(1) \setminus P'^{-1}(1)$ is at most $2^{\delta(t)}$. The probability that the tuple of literals is assigned a value in this set is less than $2^{\delta(t)} e^{-t\epsilon(t)} = \min((2/e)^{\delta(t)}, 2^{\delta(t)} e^{-\sqrt{t}}) \in o_t(1)$. Thus, the probability that a constraint, which was satisfied in I, is not satisfied in I' is $o_t(1)$ and thereby the claim has been shown. □

Now, let us turn to the second part of the proof. Let P' belong to $N_t(P, \gamma(t))$. Assume that there is an algorithm B' that approximates Max CSP(P') within β. We then show that there is an algorithm B that approximates Max CSP(P) within $\beta - o_t(1)$.

Given an instance I of Max CSP(P), B does as follows. For each constraint C_j in I, append all possible negation masks of $t-k$ auxiliary variables, $y_1, \ldots y_{t-k}$, thereby creating 2^{t-k} constraints with the same weight as C_j. These constraints constitute an instance I' of Max CSP(P') which can be approximated by B'. B runs B' on I' and returns the solution returned by B', but without the auxiliary variables. The optimal solution of I, with value w_{opt}, will yield value at least $w_{\text{opt}}(2^{t-k} - \gamma(t))$ on I' (for an arbitrary assignment to the auxiliary variables). Thus, B' will produce a solution with at least value $\beta w_{\text{opt}}(2^{t-k} - \gamma(t))$.

Let us consider a constraint C_j in I with weight w_j that was not satisfied by algorithm B. Then, algorithm B' will satisfy at most $\gamma(t)$ of the corresponding constraints in I'. If C_j instead was satisfied by B, then B' can possibly satisfy all 2^{t-k} corresponding constraints in I'. Assume that the solution returned by B satisfies constraints of total weight w_B. Then

$$\begin{aligned} \beta w_{\text{opt}}(2^{t-k} - \gamma(t)) &\leq (w_{\text{tot}} - w_B)\gamma(t) + w_B 2^{t-k} \\ &= w_{\text{tot}}\gamma(t) + w_B(2^{t-k} - \gamma(t)), \end{aligned}$$

where w_{tot} is the total weight of constraints in I. Dividing the left and right expression with $(2^{t-k} - \gamma(t))$ we get

$$w_B \;\geq\; \beta w_{\text{opt}} - w_{\text{tot}} \frac{\gamma(t)}{2^{t-k} - \gamma(t)} \;=\; \beta w_{\text{opt}} - w_{\text{tot}} \cdot o_t(1).$$

We know that β is at least the constant $\alpha_R(P)$, and w_{opt} is at least a constant fraction of w_{tot}, thus

$$w_B \;\geq\; (\beta - o_t(1)) w_{\text{opt}}.$$

We conclude that B approximates Max CSP(P) within $\beta - o_t(1)$. $\square$

7.3 Final Remarks

In this chapter we have considered the structure of approximation resistant predicates. We have shown that it is not true in general that if a predicate is implied by an approximation resistant predicate, it is approximation resistant as well. We have also shown that making very small adjustments to a predicate P, by increasing the arity and removing accepting inputs, never makes it much harder to approximate Max CSP(P). If instead some accepting inputs are added, we see that predicates can be made approximation resistant. Thus, every predicate has close neighbor predicates that are approximation resistant. This exhibits a quite interesting asymmetry between the lower and upper neighborhood of a predicate.

Chapter 8

Approximating Max kCSP

In previous chapters we have either shown that we can or cannot beat a random assignment. However, we have not really been interested in the exact approximation ratio. In this chapter we focus on this and design an algorithm in order to achieve a good approximation ratio for Max kCSP.

We start by giving some background information about Max kCSP. In Section 8.2 we present the underlying ideas of our algorithm. After that we give a formal description of the algorithm along with an analysis of its approximation ratio. In Section 8.4 we evaluate our algorithm numerically and give approximation ratios for Max kCSP, $k = 5 \dots 100$. In the last section we identify a PCP class as a subset of P.

8.1 Background

Trevisan [37] used a linear relaxation algorithm in order to 2^{1-k}-approximate Max kCSP. He observed that the hardest instances consist of only conjunctions of literals, where a literal is a variable or a negated variable. The problem Max kConjSAT consists of such instances. Trevisan used a linear relaxation algorithm in order to 2^{1-k}-approximate Max kConjSAT. We note that a random assignment satisfies a single conjunction of length k with probability 2^{-k}. This implies that a random assignment 2^{-k}-approximates Max kConjSAT and the Trevisan algorithm therefore outperforms a random assignment with a factor of two. Recently, Hast [18] produced a $2^{1.54-k}$-approximation of Max kCSP by utilizing a semidefinite relaxation approach. Essentially, it combined already known algorithms for Max 2ConjSAT, Max 3ConjSAT and Max 4ConjSAT with a technique to reduce large conjunctions into smaller ones.

If a Max kCSP instance is known to be satisfiable, then it can be $(k+1)2^{-k}$-approximated by using an algorithm by Trevisan [38]. The technique by Trevisan reduces constraints, with at most k accepting inputs, into linear constraints. A random assignment is then picked from the set of assignments that adhere to the produced linear constraints. This ensures that each constraint is accepted with probability at least $(k+1)2^{-k}$, and thus the algorithm is a $(k+1)2^{-k}$-approximation.

In this chapter we show that it is possible to $c_0 k(\log k)^{-1}2^{-k}$-approximate Max kCSP, for a constant $c_0 > 0$, even if the instance is not satisfiable. Our algorithm is the first that outperforms a random assignment with an increasing factor for larger values of k. It is interesting to see that we can match, up to a logarithmic factor, the approximation ratio of Trevisan for satisfiable instances, even though we use very different methods.

Due to the connection between PCPs and approximability of Max CSPs we have that the PCP of Samorodnitsky and Trevisan [35] and the enhancement of Engebretsen and Holmerin [7] shows that it is NP-hard to approximate Max kCSP within $2^{\sqrt{2k-2}+1/2-k}$. Our algorithm implies the following inclusion:

$$\mathrm{PCP}_{c,s}[\log, k] \subseteq \mathrm{P}, \quad \text{for any } c/s > \frac{\log k}{c_0 k} 2^k \quad ,$$

where c is the completeness and s the soundness of a verifier that uses a logarithmic number of random bits and asks k questions.

8.2 Our Method

An instance of Max kAllEqual consists of a collection of weighted constraints. Each constraint is a k-tuple of literals and a constraint is satisfied if all its literals have the same value, i.e., all are true or all are false. A simple reduction shows that an r-approximation of Max kAllEqual can be turned into an $r/2$-approximation of Max kConjSAT, and thus also into an $r/2$-approximation of Max kCSP. Here we approximate Max kAllEqual in two steps. The first step is to produce an unbalanced solution. For such a solution, constraints tend to either have many literals that are true or many that are false. We find such an unbalanced solution by using semidefinite relaxation techniques [4, 44]. In the second step we produce an assignment that is biased towards this solution. This is done by, for some $\alpha \in [0, 1]$, assigning a variable according to the unbalanced solution with probability $(1+\alpha)/2$, and negating the value with probability $(1-\alpha)/2$.

It can be shown that as long as α is chosen appropriately, such a biased random assignment makes the value of all literals in a constraint equal with much higher probability than if an unbiased random assignment is used. Too see why it is like this we look at a constraint consisting of eight literals of which the first seven have the same value but the last one has the opposite value according to the unbalanced solution. A random assignment makes all literals equal with probability $2^{-7} \approx 0.0078$. For a biased random assignment the constraint is satisfied if we assign the first seven literals according to the unbalanced solution but the last literal is negated. This happens with probability $\left(\frac{1+\alpha}{2}\right)^7\left(\frac{1-\alpha}{2}\right)$. By choosing $\alpha = 3/4$ this probability is larger than 0.049 making it more than six times more probable to satisfy the constraint using a biased random assignment compared with using an unbiased random assignment.

Let us give an indication from where the approximation ratio of our Max kAllEqual algorithm comes from. Assume that we have a Max kAllEqual instance and let w_{tot} be the total weight of its constraints and let w_{opt} be the value of an optimal solution to the instance. The expected value of a random assignment is $2^{-k}w_{\text{tot}}$, thus if $w_{\text{opt}} \leq w_{\text{tot}}/k$ then a random assignment achieves an approximation ratio of roughly $k2^{-k}$. If $w_{\text{opt}} > w_{\text{tot}}/k$ then the optimal solution turns out to be unbalanced in our measure. If we pick an assignment that is biased towards this solution we achieve an approximation ratio of $k2^{-k}$. However, we do not know the optimal solution and instead we find an unbalanced solution using the approximation algorithm by Charikar and Wirth [4], described in Section 2.7. In this process we loose a factor of $\log k$ and thus the approximation ratio gets to be $\Omega\left(k(\log k)^{-1}2^{-k}\right)$.

8.3 Algorithm Description

If some of the constraints of a Max kAllEqual are not of size k, then these constraints are padded into size k using auxiliary variables, where each new variable only appears once in the instance. This will not effect the satisfiability of the instance. Thus, we can assume that all constraints are of length k. The algorithm is shown in Figure 8.1.

Input: A set of Boolean variables $\{x_1, \ldots x_n\}$ and a set of all equal constraints $\{C_1, \ldots C_m\}$ with corresponding weights $\{w_1, \ldots w_m\}$.

1. **(2AllEqual-gadget)** Each constraint $C_i = z_{i_1} \equiv z_{i_2} \ldots \equiv z_{i_k}$ is transformed into $k(k-1)/2$ equality constraints, $z_{i_1} \equiv z_{i_2}$, $z_{i_1} \equiv z_{i_3}$, $\ldots z_{i_{k-1}} \equiv z_{i_k}$. Each constraint is given weight w_i.
2. **(Solve Max 2AllEqual)** Use the Charikar and Wirth [4] algorithm in order to satisfy as much weight of the equality constraints as possible. Let b_i be the value of variable x_i in the produced solution.
3. **(Biased random assignment)** For $i := 1, \ldots, n$: assign x_i according to

$$x_i := \begin{cases} b_i & \text{with probability } (1+\alpha)/2 \\ \bar{b}_i & \text{with probability } (1-\alpha)/2 \end{cases},$$

with $\alpha = 1/\sqrt{k}$.

Figure 8.1: Algorithm AllEq: a Max kAllEqual algorithm.

Algorithm Analysis

Theorem 8.1. *There exists a constant $c > 0$ such that* Max kAllEqual*, for $k \geq 2$, can in probabilistical polynomial time be approximated within a factor of $ck(\log k)^{-1}2^{-k}$.*

The proof of Theorem 8.1 is based on Lemmas 8.2 and 8.3.

Given an assignment to a Max kAllEqual instance we call a constraint unbalanced if either many literals are true or many literals are false. Given an assignment and a constraint C_i, γ_i is defined such that $k/2 + \gamma_i$ literals are true and $k/2 - \gamma_i$ are false. Thus, γ_i is a function depending on an assignment but due to notational convenience we do not make this dependency explicit. For a fixed assignment, we let γ_i^2 be a measure of how unbalanced C_i is. The following lemma shows that if there exists a good solution to a Max kAllEqual instance, then we are able to find an assignment which makes constraints of large weight unbalanced. We do this by first transforming the Max kAllEqual instance into a Max 2AllEqual instance, and then solving it using the Charikar-Wirth algorithm.

Lemma 8.2. *Let I be an instance of* Max kAllEqual *with total weight w_{tot} and let the normalized value of an optimal solution be $\hat{w}_{\text{opt}} = w_{\text{opt}}/w_{\text{tot}}$. Assume that $k \cdot \hat{w}_{\text{opt}} \geq 3$. Let $d = c_{\text{cw}}(k \cdot \hat{w}_{\text{opt}} - 1)/(4 \log k)$, where c_{cw} is the positive constant in Theorem 2.23. Then a solution can be produced in polynomial time such that*

$$dk \quad \leq \quad \mathrm{E}\left[\frac{1}{w_{\text{tot}}} \sum_{i:|\gamma_i|>\sqrt{k}/2} w_i \gamma_i^2\right] , \tag{8.1}$$

where the expectation is taken over the random choices of the Charikar-Wirth algorithm.

Proof. Each constraint of k literals is transformed into $k(k-1)/2$ equality constraints of arity two. If the original constraint is satisfied by an assignment, then all new equality constraints are satisfied as well by the same assignment. If $k/2+\gamma$ literals are true and $k/2-\gamma$ false, then the number of equality constraints that are not satisfied is $(k/2+\gamma)(k/2-\gamma) = k^2/4 - \gamma^2$. Thus, at least $k(k-1)/2 - k^2/4$ constraints are satisfied for any assignment.

A Max kAllEqual instance is transformed into a Max 2AllEqual instance according to the first step of Algorithm AllEq. An optimal solution that satisfies weight w_{opt} in the original Max kAllEqual instance then satisfies constraints of weight at least

$$\hat{w}_{\text{opt}}\binom{k}{2}w_{\text{tot}} + (1-\hat{w}_{\text{opt}})\left(\binom{k}{2} - \frac{k^2}{4}\right)w_{\text{tot}} \quad = \quad \left(\binom{k}{2} - \frac{k^2}{4} + \hat{w}_{\text{opt}}\frac{k^2}{4}\right)w_{\text{tot}} \ ,$$

in the Max 2AllEqual instance. A solution with value $W_{eq}(1/2+\delta)$ has gain δ, where $W_{eq} = w_{\text{tot}}k(k-1)/2$ is the total weight of the Max 2AllEqual instance.

The gain is a measure of how much better a solution is compared to a random assignment. We let δ^* denote the optimal gain. We then have

$$W_{eq}\left(\frac{1}{2}+\delta^*\right) \quad \geq \quad \left(\binom{k}{2}-\frac{k^2}{4}+\hat{w}_{\text{opt}}\frac{k^2}{4}\right)w_{\text{tot}} \ . \tag{8.2}$$

We derive a lower bound for the optimal gain by first subtracting $W_{eq}/2$ on both sides of (8.2) and then dividing with w_{tot},

$$\begin{aligned}\frac{k(k-1)}{2}\delta^* \quad &\geq \quad \frac{k(k-1)}{2}-\frac{k^2}{4}+\hat{w}_{\text{opt}}\frac{k^2}{4}-\frac{k(k-1)}{4} \\ &= \quad \hat{w}_{\text{opt}}\frac{k^2}{4}-\frac{k}{4} \ .\end{aligned}$$

We divide with $k(k-1)/2$ on both sides and get

$$\delta^* \quad \geq \quad \frac{k\cdot\hat{w}_{\text{opt}}-1}{2(k-1)} \ .$$

This lower bound is used in order to obtain a performance guarantee of the Charikar-Wirth algorithm from Lemma 2.23. We analyze the logarithmic factor, $\log(1/\delta^*)$, that is lost when using the Charikar-Wirth algorithm. The assumption from Lemma 8.2 implies that $k\cdot\hat{w}_{\text{opt}}-1\geq 2$, thus

$$\begin{aligned}\log(1/\delta^*) \quad &\leq \quad \log\frac{2(k-1)}{k\cdot\hat{w}_{\text{opt}}-1} \\ &< \quad \log k \ .\end{aligned}$$

We run the algorithm of Charikar and Wirth. Lemma 2.23 implies that the expected gain δ of the produced solution can be lower bounded

$$\begin{aligned}\delta \quad &\geq \quad c_{\text{cw}}\frac{\delta^*}{\log(1/\delta^*)} \\ &> \quad c_{\text{cw}}\frac{k\cdot\hat{w}_{\text{opt}}-1}{2(k-1)\log k} \ . \end{aligned}\tag{8.3}$$

We let γ_i relate to this assignment, thus γ_i is the value such that $k/2+\gamma_i$ of the literals in constraint C_i are true and $k/2-\gamma_i$ are false according to the solution produced by the Charikar-Wirth algorithm. The weight of satisfied equality constraints corresponding to constraint C_i is then

$$\left(\binom{k}{2}-\left(\frac{k}{2}+\gamma_i\right)\left(\frac{k}{2}-\gamma_i\right)\right)w_i \quad = \quad \left(\frac{k(k-1)}{4}+\left(\gamma_i^2-\frac{k}{4}\right)\right)w_i \ .$$

We sum the contribution of each constraint in order to get the total weight of satisfied equality constraints. Thus,

$$W_{eq}\left(\frac{1}{2}+\delta\right) \quad = \quad \mathrm{E}\left[\sum_{i=1}^{m}w_i\left(\frac{k(k-1)}{4}+\left(\gamma_i^2-\frac{k}{4}\right)\right)\right]$$

which implies that

$$\binom{k}{2}\delta w_{\text{tot}} = \mathrm{E}\left[\sum_{i=1}^{m} w_i\left(\gamma_i^2 - \frac{k}{4}\right)\right] . \tag{8.4}$$

We derive from (8.3) a lower bound for the weighted sum of γ_i^2 which concludes the proof of the lemma:

$$\begin{aligned}
\mathrm{E}\left[\frac{1}{w_{\text{tot}}}\sum_{i:|\gamma_i|>\sqrt{k}/2} w_i\gamma_i^2\right] &\geq \mathrm{E}\left[\frac{1}{w_{\text{tot}}}\sum_{i:|\gamma_i|>\sqrt{k}/2} w_i\left(\gamma_i^2-\frac{k}{4}\right)\right]\\
&\geq \mathrm{E}\left[\frac{1}{w_{\text{tot}}}\sum_{i=1}^{m} w_i\left(\gamma_i^2-\frac{k}{4}\right)\right]\\
&= \binom{k}{2}\delta\\
&\geq \frac{k(k-1)}{2}c_{\text{cw}}\frac{k\cdot\hat{w}_{\text{opt}}-1}{2(k-1)\log k}\\
&\geq \frac{kc_{\text{cw}}(k\cdot\hat{w}_{\text{opt}}-1)}{4\log k} ,
\end{aligned}$$

where the equality follows from (8.4). □

Lemma 8.2 shows that we can find an unbalanced solution. The following lemma shows that we can produce a good solution from such an unbalanced solution.

Lemma 8.3. *Let I be an instance of* Max kAllEqual *with total weight w_{tot} and $\{b_1,\ldots b_n\}$ is a solution such that*

$$dk \leq \frac{1}{w_{\text{tot}}}\sum_{i:|\gamma_i|>\sqrt{k}/2} w_i\gamma_i^2 ,$$

for $d \geq 1$. Then assigning

$$x_i := \begin{cases} b_i & \text{with probability } (1+\alpha)/2\\ \bar{b}_i & \text{with probability } (1-\alpha)/2 \end{cases} ,$$

with $\alpha = 1/\sqrt{k}$, produces a solution with expected value of at least $e^{2\sqrt{d}-1/2}2^{-k}w_{\text{tot}}$.

Proof. Consider a constraint C_i that has $|\gamma_i| > \sqrt{k}/2$. Let AE_i be the event that all literals in C_i get the same value after the biased random assignment.

$$\begin{aligned}
\Pr[AE_i] &= \Pr[\text{all literals true}] + \Pr[\text{all literals false}] \\
&= \left(\frac{1+\alpha}{2}\right)^{\frac{k}{2}+\gamma_i}\left(\frac{1-\alpha}{2}\right)^{\frac{k}{2}-\gamma_i} + \left(\frac{1-\alpha}{2}\right)^{\frac{k}{2}+\gamma_i}\left(\frac{1+\alpha}{2}\right)^{\frac{k}{2}-\gamma_i} \\
&> 2^{-k}\left((1+\alpha)^{\frac{k}{2}+|\gamma_i|}(1-\alpha)^{\frac{k}{2}-|\gamma_i|}\right) \\
&= 2^{-k}\left((1+\alpha)^{2|\gamma_i|}\left(1-\alpha^2\right)^{\frac{k}{2}-|\gamma_i|}\right) \\
&> 2^{-k}e^{2\alpha|\gamma_i|-\frac{k}{2}\alpha^2} \quad .
\end{aligned}$$

The following claim validates the last inequality.

Claim 8.4. *For $1 \geq \alpha > 0$ and $2|\gamma_i| \geq k\alpha$,*

$$(1+\alpha)^{2|\gamma_i|}\left(1-\alpha^2\right)^{\frac{k}{2}-|\gamma_i|} > e^{2\alpha|\gamma_i|-\frac{k}{2}\alpha^2} \quad .$$

Proof. We show that the logarithm of the left hand side, LH, is larger than the logarithm of the right hand side, RH.

$$\begin{aligned}
\ln(LH) &= 2|\gamma_i|\ln(1+\alpha) + \left(\frac{k}{2}-|\gamma_i|\right)\ln\left(1-\alpha^2\right) \\
&= 2|\gamma_i|\left(\alpha - \frac{\alpha^2}{2} + \frac{\alpha^3}{3} - \frac{\alpha^4}{4} + \ldots\right) + \\
&\quad \left(\frac{k}{2}-|\gamma_i|\right)\left(-\alpha^2 - \frac{\alpha^4}{2} - \frac{\alpha^6}{3} - \ldots\right) \\
&= 2|\gamma_i|\left(\alpha + \frac{\alpha^3}{3} + \frac{\alpha^5}{5} + \ldots\right) - \frac{k}{2}\left(\alpha^2 + \frac{\alpha^4}{2} + \frac{\alpha^6}{3} + \ldots\right) \\
&= 2\alpha|\gamma_i| - \frac{k}{2}\alpha^2 + 2|\gamma_i|\left(\frac{\alpha^3}{3} + \frac{\alpha^5}{5} + \ldots\right) - k\alpha\left(\frac{\alpha^3}{4} + \frac{\alpha^5}{6} + \ldots\right) \\
&> 2\alpha|\gamma_i| - \frac{k}{2}\alpha^2 + (2|\gamma_i| - k\alpha)\left(\frac{\alpha^3}{3} + \frac{\alpha^5}{5} + \ldots\right) \\
&\geq 2\alpha|\gamma_i| - \frac{k}{2}\alpha^2 \\
&= \ln(RH)
\end{aligned}$$

□

We calculate a lower bound on the expected weight of constraints that either have all literals true or all literals false.

$$\sum_{i:|\gamma_i|>\sqrt{k}/2} w_i \Pr[AE_i] \; > \; 2^{-k}\sum_{i:|\gamma_i|>\sqrt{k}/2} w_i e^{2\alpha|\gamma_i|-\frac{k}{2}\alpha^2} \tag{8.5}$$

We let $s_i = \gamma_i^2$ and rewrite the above lower bound:

$$2^{-k} e^{-\frac{k}{2}\alpha^2} \sum_{i:\sqrt{s_i}>\sqrt{k}/2} w_i e^{2\alpha\sqrt{s_i}} \ .$$

We consider the terms $e^{2\alpha\sqrt{s_i}}$ as functions of s_i and calculate their second derivative:

$$\frac{\partial}{\partial^2 s_i} e^{2\alpha\sqrt{s_i}} \quad = \quad \left(\alpha - \frac{1}{2\sqrt{s_i}}\right) \alpha s_i^{-1} e^{2\alpha\sqrt{s_i}} \ .$$

We see that the second derivative is positive as long as $\alpha > 1/2\sqrt{s_i}$, thus $e^{2\alpha\sqrt{s_i}}$ is convex for all terms of the sum because $\sqrt{s_i} > \sqrt{k}/2$ and $\alpha = 1/\sqrt{k}$. The condition in the lemma gives a lower bound to $\sum_{i:\sqrt{s_i}>\sqrt{k}/2} w_i s_i$. Thus, using Jensen's inequality, Proposition 2.2, we conclude that (8.5) is minimized if all values of $|\gamma_i|$ over the threshold are equal.

We let W_+ be the weight of all conjunctions meeting the threshold condition, $W_+ = \sum_{i:|\gamma_i|>\sqrt{k}/2} w_i$. We will see that the worst case happens if $W_+ = w_{\text{tot}}$, but for now we are general and let $W_+ = x w_{\text{tot}}$ where x is a value between 0 and 1. In order to minimize (8.5), the value of all $|\gamma_i|$ above the threshold should be equal and by the assumption of the lemma at least $\sqrt{dk/x}$. We apply this to (8.5) and get the expected weight of constraints that either have all literals true or all literals false:

$$2^{-k} \sum_{i:|\gamma_i|>\sqrt{k}/2} w_i e^{\frac{2|\gamma_i|}{\sqrt{k}} - \frac{1}{2}} \geq 2^{-k} x w_{\text{tot}} e^{2\sqrt{d/x}-1/2} \ . \tag{8.6}$$

We calculate its derivative in order to minimize the above expression with respect to x:

$$\frac{\partial}{\partial x} 2^{-k} x w_{\text{tot}} e^{2\sqrt{d/x}-1/2} \quad = \quad (1-\sqrt{d/x}) 2^{-k} e^{2\sqrt{d/x}-1/2} w_{\text{tot}} \ .$$

As $d \geq 1$, we see that the derivative is non-positive for $x \in (0,1]$. Thus, (8.6) is minimized by setting $x = 1$ and then the expected weight of satisfied constraints is at least $2^{2\sqrt{d}-1/2} 2^{-k} w_{\text{tot}}$ which concludes the proof of Lemma 8.3. $\square$

We are now ready to prove Theorem 8.1 by using Lemma 8.2 and Lemma 8.3.

Proof. Let I be an instance of Max kAllEqual with total weight w_{tot} and where $\hat{w}_{\text{opt}} \cdot w_{\text{tot}}$ is the value of an optimal solution. Set $c = (c_1/c_{\text{cw}} + 1)^{-1}$, where c_1 is a positive constant yet to be defined. If $\hat{w}_{\text{opt}} \leq \log k/(ck)$, then a random assignment achieves the following approximation ratio:

$$\frac{2^{-k} w_{\text{tot}}}{\log k/(ck) w_{\text{tot}}} \quad = \quad \frac{ck}{\log k} 2^{-k} \ .$$

Thus, we only need to consider if $\hat{w}_{\text{opt}} > \log k/(ck)$. Therefore, assume that $\hat{w}_{\text{opt}} = r \log k/(ck)$ for some $r > 1$. We note that $k \cdot \hat{w}_{\text{opt}} \geq \log k/c > 3$ and apply Lemma 8.2 with

$$d = c_{\text{cw}} \frac{k \cdot \hat{w}_{\text{opt}} - 1}{4 \log k} = c_{\text{cw}} \frac{r/c - 1}{4} = c_{\text{cw}} \frac{r(c_1/c_{\text{cw}} + 1) - 1}{4} > rc_1/4 \ .$$

We can now apply Lemma 8.3 with an expected value of $d = rc_1/4$. However, if $d < 1$ then Lemma 8.3 does not give anything and thus the expected value of the solution could be zero. The following function expresses a lower bound for the expected value of the solution

$$\begin{cases} 0 & \text{if } 0 \leq d < 1 \\ e^{2\sqrt{d}-1/2} 2^{-k} w_{\text{tot}} & \text{if } d \geq 1 \end{cases} \ . \tag{8.7}$$

It is not convex and thus we cannot apply Jensen's inequality. However, the following linear function is a lower bound for (8.7):

$$c_2(d-1)2^{-k} w_{\text{tot}} \ , \quad \text{where} \quad c_2 = \min_{d>1} \frac{e^{2\sqrt{d}-1/2}}{d-1} \approx 9.5 \ . \tag{8.8}$$

By applying the lower bound (8.8) we get that w_{app}, the expected value of the solution, is at least $c_2(rc_1/4 - 1)2^{-k} w_{\text{tot}}$. By setting $c_1 = 4/c_2 + 4$ we ensure that $w_{\text{app}} \geq r2^{-k} w_{\text{tot}}$ and thus the approximation ratio is

$$\frac{w_{\text{app}}}{w_{\text{opt}}} \geq \frac{r2^{-k} w_{\text{tot}}}{r \log k/(ck) w_{\text{tot}}} = \frac{ck}{\log k} 2^{-k} \ ,$$

which concludes the proof of Theorem 8.1. □

Consequences for Max kConjSAT and Max kCSP

By using a simple observation, we derive from Theorem 8.1 the following result on the approximability of Max kConjSAT.

Theorem 8.5. *There exists a constant $c_0 > 0$ such that* Max kConjSAT, *for $k \geq 2$, can in probabilistical polynomial time be approximated within a factor of $c_0 k (\log k)^{-1} 2^{-k}$.*

Proof. Assume that I is an Max kConjSAT instance with optimal value w_{opt}. In order to approximate I we regard it is a Max kAllEqual instance I' and run Algorithm AllEq. As the optimal solution value of I' is at least as large as w_{opt}, the expected value of the solution is according to Theorem 8.1 at least $ck(\log k)^{-1} 2^{-k} w_{\text{opt}}$. If the assignment satisfies constraints of more weight due to that all literals are false rather than all literals are true, then the assignment is negated. This does not effect the objective value but ensures that at least half the weight of satisfied constraints are satisfied due to that all literals are true. Thus, the proof is concluded by observing that the expected value of the assignment evaluated on I is at least $c_0 k (\log k)^{-1} 2^{-k} w_{\text{opt}}$, where $c_0 = c/2$. □

By using the observation of Trevisan [37], we conclude that an algorithm for Max kConjSAT implies an approximation algorithm for Max kCSP with the same approximation ratio. Thus, as a consequence of Theorem 8.5 we get our main theorem:

Theorem 8.6. *There exists a constant* $c_0 > 0$ *such that* Max kCSP, *for* $k \geq 2$, *can in probabilistical polynomial time be* $c_0 k(\log k)^{-1} 2^{-k}$*-approximated.*

8.4 Numerical Approximation Ratios

In this section, we give approximation ratios for Max kCSP for values of $5 \leq k \leq 100$. The ratios are obtained using numerical methods and are presented in Figure 8.2. For $k \leq 4$ there are known good approximation algorithms, which outperform our algorithm with a broad margin [15, 31, 42].

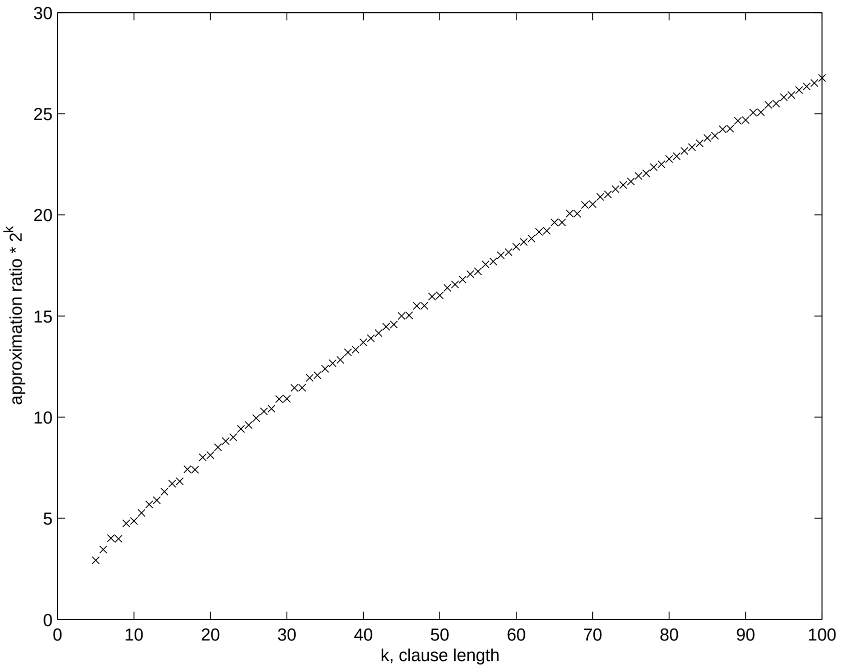

Figure 8.2: Approximation ratios on Max kCSP.

In this section we use an algorithm slightly different from Algorithm AllEq which is presented in Figure 8.1. First we use Zwick's outward rotation algorithm [44] in order to approximate the Max 2AllEqual instance instead of the Charikar-Wirth algorithm. This choice is made because an exact approximation ratio can be calculated numerically from [44], as long as the optimal gain is known. In the last step of Algorithm AllEq, we use a somewhat different value of the bias α. Given the solution of the Max 2AllEqual instance, it is possible to calculate the optimal value of the parameter α. However, our calculation shows that by setting

$$\alpha = \sqrt{\frac{1}{k} + \left(2 - \frac{2}{k}\right)\delta} \ ,$$

where δ is the expected gain of the approximate solution, we obtain an approximation ratio very close to the ratio that would be obtained if we used the optimal value of α. We choose this suboptimal value to ensure easier reproducibility of the graph in Figure 8.2.

We can heuristically show why this value of α is suitable. It seems that the worst case is if the solution makes all constraints equally unbalanced, thus the value of $|\gamma_i|$ is equal for all constraints. In the proof of Lemma 8.2 we saw that

$$\binom{k}{2}\delta w_{\text{tot}} \quad = \quad \mathrm{E}\left[\sum_{i=1}^{m} w_i \left(\gamma_i^2 - \frac{k}{4}\right)\right] \ .$$

This implies that the worst case is if

$$|\gamma_i| = \sqrt{\frac{k(k-1)}{2}\delta + \frac{k}{4}} \ .$$

In the proof of Lemma 8.3 we showed that the probability that all literals in a constraint get the same value after a biased random assignment, is at least $2^{-k}e^{2\alpha|\gamma_i| - \frac{k}{2}\alpha^2}$. It is easy to show that this expression is maximized by setting $\alpha = 2|\gamma_i|/k$. Inserting the worst case value for $|\gamma_i|$ in this expression, we see that α is equal to the value we use.

For each value of k, we calculate the approximation ratio in the following way: For each possible optimal gain, using a suitable discretization, we calculate the expected gain δ of the solution produced by the algorithm of Zwick. We then produce a linear program where the objective value is equal to the expected normalized objective value of the solution produced by the biased random assignment. The linear program has $k+1$ variables $y_0, y_1, \ldots y_k$, where y_i indicates the total normalized weight of constraints with exactly i literals true. There are also constants $d_0, \ldots d_k$, where $d_i = \frac{k(k-1)}{2} - i(k-i)$ is the number of pair of literals that are equal in a constraint with exactly i literals true. We let b_i be the probability that a constraint with exactly i literals true will be all equal after the biased random assignment. This value depends on our choice of α.

$$\begin{aligned} &\min \sum_{i=0}^{k} b_i y_i, \\ &\text{given} \quad \textstyle\sum_{i=0}^{k} y_i = 1 \\ &\qquad\quad\ \textstyle\sum_{i=0}^{k} d_i y_i = \frac{k(k-1)}{2}\left(\frac{1}{2} + \delta\right) \end{aligned}$$

By solving the linear program we get a numerical value of the approximation ratio of the Max kAllEqual algorithm. By dividing this value with two, we get the approximation ratio of the Max kCSP algorithm.

Our algorithm approximates Max 5CSP within $2.91 \cdot 2^{-5}$. The general $2^{1.54-k} = 2.90 \cdot 2^{-k}$ approximation algorithm in [18] does better for small values of k and actually $3.68 \cdot 2^{-5}$ approximates Max 5CSP. However, for values of $k \geq 6$ our algorithm achieves the best known approximation ratio for Max kCSP.

8.5 Relation with PCP Classes

The complexity class $\mathrm{PCP}_{c,s}[\log, q]$ contains all languages that have a verifier with completeness c, soundness s, which uses only a logarithmic number of random bits and asks at most q (adaptive) questions. Trevisan showed that there is a close connection between the power of PCPs asking k questions and the approximability of Max kConjSAT.

Theorem 8.7 (Trevisan [37]). *If, for some $r \leq 1$,* Max kConjSAT *is deterministically r-approximable in polynomial time then* $\mathrm{PCP}_{c,s}[\log, k] \subseteq \mathrm{P}$ *for any $c/s > 1/r$.*

Our Max kConjSAT algorithm is probabilistic, thus it is not immediate that the above theorem can be applied. There are two steps in our algorithm that are probabilistic: the rounding in the Charikar-Wirth algorithm and the biased random assignment. However, in Theorem 8.7 we only need an algorithm that gives a lower bound of the optimal value, not an approximate solution. We get such a lower bound directly from the semidefinite program of the Charikar-Wirth algorithm. Thus, we have the following theorem.

Theorem 8.8. $\mathrm{PCP}_{c,s}[\log, k] \subseteq \mathrm{P}$ *for any $c/s > \frac{\log k}{c_0 k} 2^k$, where $c_0 > 0$ is the constant in Theorem 8.5.*

Chapter 9

Summary and Discussion

In this thesis, we have studied the approximability of Boolean constraint satisfaction problems, Max CSPs. An instance of a Max CSP consists of a set of weighted constraints acting over a set of Boolean variables. The objective is to find an assignment to the variables such that the weight of satisfied constraints is maximized. If all constraints act over at most k different variables, then it is a Max kCSP instance. Another subproblem to Max CSP is Max CSP(P), where P is a predicate. In such an instance, the predicate P is used to determine if a constraint is satisfied or not.

9.1 Approximation Resistant Predicates

A very natural algorithm for many combinatorial optimization problems is choosing a solution uniformly at random from the solution space. For Max CSP instances this is done by assigning an unbiased and independent random value to each variable. Despite its simplicity, such a random assignment is for some Max CSP essentially the best possible efficient approximation algorithm. In this thesis we investigate for which types of Max CSP this is the case. In particular, we study for which predicates P it is NP-hard to outperform a random assignment on Max CSP(P). We call such predicates approximation resistant.

We extend PCP techniques by Håstad [22] and Samorodnisky and Trevisan [35] in order to characterize predicates as approximation resistant. We are able to show that predicates with many accepting inputs are approximation resistant.

A central predicate for these techniques is the parity predicate 3XOR. Håstad [22] and Zwick [42] established that the only predicates of arity three that are approximation resistant are predicates that only reject inputs of the same parity. We strengthen this central role of the parity function by showing that if an instance is hard, then an optimal solution makes almost all constraints have the same parity. A hard instance is an almost satisfiable instance, for which it is NP-hard to approximate it significantly better than picking a random assignment.

The objective value function of a Max CSP(P) instance is a multilinear expression. By maximizing the sum of linear and bi-linear terms in this expression, we are for some predicates P able to produce a solution that has a somewhat higher value than the expected value of a random solution. A Max CSP(3XOR) instance consists of only monomials of degree three. As 3XOR is approximation resistant, it seems impossible to extend this algorithm in optimizing degree three terms as well. A possible extension of the method is instead to use it in order to characterize almost satisfying solutions. Using the canonical relaxation of a Max CSP(P), introduced by Karloff and Zwick [28], this information could potentially enable us to make a better relaxation for approximating almost satisfiable instances.

We have both classified predicates as approximation resistant as well as being not approximation resistant. However, some predicates cannot be characterized using the methods in this thesis. The fraction of uncharacterized predicates grows when we consider predicates of larger arity. Approximately 85% of the predicates of arity four were characterized, but when the arity tends to infinity we are not even able to characterize a constant fraction of the predicates.

9.2 Approximability of Max kCSP

Another contribution of this thesis is an algorithm for approximating Max kCSP. Traditionally, approximating Max kCSP has been done by approximating Max kConjSAT in conjunction with a reduction of Max kCSP to Max kConjSAT [18, 37]. In Max kConjSAT all constraints are conjunctions of at most k literals. We instead give an algorithm for approximating Max kAllEqual, where every constraint consists of at most k literals and is satisfied if all literals have the same value. An α-approximation for Max kAllEqual can easily be transformed into an $\alpha/2$-approximation for Max kConjSAT. Therefore, our Max kAllEqual algorithm yields an algorithm for Max kCSP. The approximation ratio of $\Omega(2^{-k+\log k-\log\log k})$ is a substantial improvement to the previous best ratio of $2^{1.54-k}$ [18].

Another approach to approximate Max kCSP is to try and apply the semidefinite relaxation method more directly. The canonical semidefinite relaxation of Karloff and Zwick [28] yields the strongest possible semidefinite relaxation from a natural class of relaxations. A solution to the semidefinite program is a vector configuration $(v_i)_{i=0}^{n}$, where each vector v_i is a relaxation of a variable x_i in the original instance. A possible solution to the semidefinite program is a set of orthogonal vectors. If the original instance is a Max kConjSAT instance, it can be shown that this solution has value $(2\lceil(k+1)/2\rceil)^{-1}w_{\text{tot}}$, where w_{tot} is the sum of the weight of every constraint. However, such an orthogonal solution obviously does not give any specific information about the original instance. If an optimal solution of a Max kConjSAT instance has value less than $(2\lceil(k+1)/2\rceil)^{-1}w_{\text{tot}}$ it is thus hard to see how to use semidefinite relaxation techniques in order to obtain useful information about the instance.

It seems that the above indicates a natural limit for semidefinite relaxation methods for approximating Max kCSP. It is hard to see how such an approach can achieve an approximation ratio of $\omega(2^{-k+\log k})$. It has been shown that it is NP-hard to $2^{-k+c\sqrt{k}}$-approximate Max kCSP, for a certain constant c [7, 35]. A challenging open problem is to bridge the gap between the negative and the positive results.

Bibliography

[1] Farid Alizadeh. Interior point methods in semidefinite programming with applications to combinatorial optimization. *SIAM Journal on Optimization*, 5(1):13–51, 1995.

[2] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and hardness of approximation problems. *Journal of the ACM*, 45(3):501–555, 1998. Preliminary version appeared in FOCS 1992.

[3] Mihir Bellare, Oded Goldreich, and Madhu Sudan. Free bits, PCPs, and nonapproximability - towards tight results. *SIAM Journal on Computing*, 27(3):804–915, 1998.

[4] Moses Charikar and Anthony Wirth. Maximizing quadratic programs: Extending Grothendieck's inequality. In *Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science*, pages 54–60, 2004.

[5] Stephen A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, pages 151–158, 1971.

[6] Nadia Creignou. A dichotomy theorem for maximum generalized satisability problems. *Journal of Computer and System Sciences*, 51(3):511–522, 1995.

[7] Lars Engebretsen and Jonas Holmerin. More efficient queries in PCPs for NP and improved approximation hardness of maximum CSP. In *Proceedings of STACS 2005, Lecture Notes in Computer Science 3404*, pages 194–205, 2005.

[8] Uriel Feige. A threshold of ln n for approximating set cover. *Journal of the ACM*, 45(4):634–652, 1998.

[9] Uriel Feige and Joe Kilian. Zero knowledge and the chromatic number. In *IEEE Conference on Computational Complexity*, pages 278–287, 1996.

[10] Uriel Feige and Michael Langberg. The RPR^2 rounding technique for semidefinite programs. In *Proceedings of ICALP*, pages 213–224, 2001.

[11] Uriel Feige and Daniel Reichman. On systems of linear equations with two variables per equation. In *Proceedings of APPROX 2004, Lecture Notes in Computer Science 3122*, pages 117–127, 2004.

[12] Michael R. Garey and David S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness.* W. H. Freeman & Co., 1979.

[13] Michael R. Garey, David S. Johnson, and Larry J. Stockmeyer. Some simplified NP-complete graph problems. *Theoretical Computer Science*, 1(3):237–267, 1976.

[14] Michel X. Goemans and David P. Williamson. Improved Approximation Algorithms for Maximum Cut and Satisfiability Problems Using Semidefinite Programming. *Journal of the ACM*, 42:1115–1145, 1995.

[15] Venkatesan Guruswami, Daniel Lewin, Madhu Sudan, and Luca Trevisan. A tight characterization of NP with 3 query PCPs. In *Proceedings of the 39th Annual IEEE Symposium on Foundations of Computer Science*, pages 8–17, 1998.

[16] David J. Haglin and Shankar M. Venkatesan. Approximation and intractability results for the maximum cut problem and its variants. *IEEE Transactions on Computers*, 40(1):110–113, 1991.

[17] Eran Halperin and Uri Zwick. Approximation algorithms for MAX 4-SAT and rounding procedures for semidefinite programs. *Journal of Algorithms*, 40:185–211, 2001.

[18] Gustav Hast. Approximating Max kCSP using random restrictions. In *Proceedings of APPROX 2004, Lecture Notes in Computer Science 3122*, pages 151–162, 2004.

[19] Gustav Hast. Approximating Max kCSP - outperforming a random assignment with almost a linear factor. Accepted to ICALP 05, 2005.

[20] Gustav Hast. Beating a random assignment. Manuscript, 2005.

[21] Johan Håstad. Clique is hard to approximate within $n^{1-\epsilon}$. *Acta Mathematica*, 182:105–142, 1999.

[22] Johan Håstad. Some optimal inapproximability results. *Journal of the ACM*, 48(4):798–859, 2001. Preliminary version appeared in STOC 1997.

[23] Johan Håstad. Every 2-CSP allows nontrivial approximation. Accepted to STOC 2005, 2005.

[24] Johan Håstad and Avi Wigderson. Simple analysis of graph tests for linearity and PCP. *Random Structures and Algorithms*, 22(2):139–160, 2003.

[25] Thomas Hofmeister and Hanno Lefmann. A combinatorial design approach to MAXCUT. In *Proceedings of STACS 1996, Lecture Notes in Computer Science 1046*, pages 441–452, 1996.

[26] David S. Johnson. Approximation algorithms for combinatorial problems. *Journal of Computer and System Sciences*, 9(3):256–278, 1974.

[27] David Karger, Rajeev Motwani, and Madhu Sudan. Approximate graph coloring by semidefinite programming. *Journal of the ACM*, 45(2):246–265, 1998.

[28] Howard Karloff and Uri Zwick. A 7/8-approximation algorithm for MAX 3SAT? In *Proceedings of the 38th Annual IEEE Symposium on Foundations of Computer Science*, pages 406–415, 1997.

[29] Sanjeev Khanna, Madhu Sudan, Luca Trevisan, and David P. Williamson. The approximability of constraint satisfaction problems. *SIAM Journal on Computing*, 30(6):1863–1920, 2000.

[30] Subhash Khot, Guy Kindler, Elchanan Mossel, and Ryan O'Donnell. Optimal inapproximability results for MAX-CUT and other 2-variable CSPs? In *Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science*, pages 146–154, 2004.

[31] Michael Lewin, Dror Livnat, and Uri Zwick. Improved rounding techniques for the MAX 2-SAT and MAX DI-CUT problems. In *Proceedings of 9th IPCO, Lecture Notes in Computer Science 2337*, pages 67–82, 2002.

[32] S. Poljak and D. Turzík. A polynomial algorithm for constructing a large bipartite subgraph, with an application to a satisfiability problem. *Canadian Journal of Mathematics*, 34:519–524, 1982.

[33] Ran Raz. A parallel repetition theorem. *SIAM Journal on Computing*, 27(3):763–803, 1998.

[34] Sartaj Sahni and Teofilo F. Gonzalez. P-complete approximation problems. *Journal of the ACM*, 23(3):555–565, 1976.

[35] Alex Samorodnitsky and Luca Trevisan. A PCP characterization of NP with optimal amortized query complexity. In *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing*, pages 191–199, 2000.

[36] Thomas Schaefer. The complexity of satisability problems. In *Proceedings of the 10th Annual ACM Symposium on Theory of Computing*, pages 216–226, 1978.

[37] Luca Trevisan. Parallel approximation algorithms by positive linear programming. *Algorithmica*, 21(1):72–88, 1998.

[38] Luca Trevisan. Approximating satisfiable satisfiability problems. *Algorithmica*, 28(1):145–172, 2000.

[39] Luca Trevisan, Gregory B. Sorkin, Madhu Sudan, and David P. Williamson. Gadgets, approximation, and linear programming. *SIAM Journal on Computing*, 29:2074–2097, 2000.

[40] Alan M. Turing. On computable numbers with an application to the entscheidungs problem. *Proceedings of the London Mathematical Society*, 42:23–265, 1936.

[41] P. M. Vitányi. How well can a graph be n-colored? *Discrete Mathematics*, 34:69–80, 1981.

[42] Uri Zwick. Approximation algorithms for constraint satisfaction problems involving at most three variables per constraint. In *Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 201–210, 1998.

[43] Uri Zwick. Finding almost satisfying assignments. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing*, pages 551–560, 1998.

[44] Uri Zwick. Outward rotations: a tool for rounding solutions of semidefinite programming relaxations, with applications to MAX CUT and other problems. In *Proceedings of the 31st Annual ACM Symposium on Theory of Computing*, pages 679–687, 1999.