Constraint Satisfaction Problems
Lecture Notes

Yuval Filmus

June 12, 2022

Contents

1 Introduction

1.1 What are CSPs?

A *constraint satisfaction problem* consists of a collection of *variables* and *constraints*. Each variable has an associated *domain*. The corresponding problem asks whether there is an assignment to the variables that satisfies all constraints; though this is not the only interesting problem which one can ask about a CSP instance. We start with two examples: SAT and coloring.

SAT SAT is the prototypical NP-complete problem. As usually described, the input is a CNF, that is, a conjunction of clauses, each of which is a disjunction of literals; and the desired output is whether the CNF has a satisfying assignment. We can think of the various clauses as constraints. For example, the clause $x \vee y \vee z$ states that one of the variables x, y, z must be assigned *True*. The instance is satisfiable if some truth assignment satisfies all constraints.

k-coloring Given a graph and an integer k, the k-coloring problem asks whether the graph G has a legal coloring using k colors. When k is fixed, we can model this as a CSP whose variables are the colors of the vertices $c(v) \in \{1, \ldots, k\}$. Each edge $\{u, v\}$ gives us a constraint $c(u) \neq c(v)$. The graph is k-colorable if some coloring satisfies all these constraints.

Other well-known problems also correspond to instances of CSPs, but the problem itself is different. Here are three examples.

MAX-SAT Given a CNF, the goal is to find a truth assignment that satisfies as many clauses as possible. More generally, given a CSP instance, one can ask for a variable assignment that satisfies as many constraints as possible.

MAX-CUT Given a graph, the goal is to find a partition of the vertex set into two parts that cuts the maximum number of edges. The corresponding CSP instance has a Boolean variable $c(v)$ for each vertex, signifying which part v belongs to, and a constraint $c(u) \neq c(v)$ for each edge $\{u, v\}$. The goal is to find a variable assignment which satisfies the maximal number of constraints.

Vertex Cover Given a graph, the goal is to find a subset of vertices which is incident to all edges. The corresponding CSP instance has a Boolean variable $c(v)$ for each vertex, and a constraint $c(u) \vee c(v)$ for each edge $\{u, v\}$. The goal is to find a variable assignment of lowest weight which satisfies all constraints.

In all problems listed above, the variables are restricted to a particular domain, and the constraints are of a particular form, namely, they come from a particular *inventory* of constraints:

- In SAT and MAX-SAT, the variables are Boolean (such problems are known as *Boolean CSPs*), and the constraints are of the form $x_{i_1} \vee \cdots \vee x_{i_a} \vee \bar{x}_{j_1} \vee \cdots \vee \bar{x}_{j_b}$.
- In k-coloring and in MAX-CUT, the variables have the domain $\{1, \ldots, k\}$ (where $k = 2$ in MAX-CUT), and the only allowed constraints are of the form $x_i \neq x_j$.
- In Vertex Cover, the variables are Boolean, and the constraints are of the form $x_i \vee x_j$.

There are other ways to restrict CSP instances. For example, we could require that every variable appear at most (or exactly) so many times, or that the variable-constraint graph satisfy some property (say, it is planar). We won't consider such restrictions in this course. Other restrictions, which might show up, are on the solution space. For example, we might ask what happens if there is at most one satisfying assignment.

1.2 Some questions

Given a CSP instance, here are some questions we might be interested in:

- Is the instance satisfiable?
- How many constraints can we satisfy simultaneously?
- What is the minimum weight of a satisfying assignment?
- How many satisfying assignments are there?
- What is the parity of the number of solutions?
- If the instance is satisfiable, can we sample a satisfying assignment?
- If the instance is not satisfiable, can we prove this?

The complexity of these questions depends on the allowed constraints. Another line of research asks what happens for random instances, suitably defined; this research direction is related to ideas from statistical mechanics.

We will concentrate on the case of Boolean CSPs in which the allowed constraints come from a finite set $\mathcal{I}$. Each element of $\mathcal{I}$ is a predicate $P \subseteq \{0,1\}^k$. A constraint is an *instance* of P if it is of the form $(x_{i_1}, \ldots, x_{i_k}) \in P$, where $x_{i_1}, \ldots, x_{i_k}$ are arbitrary variables, not necessarily distinct. We denote the collection of all such CSPs by CSP($\mathcal{I}$).

For example, 3SAT is CSP($\{P_0, P_1, P_2, P_3\}$), where

$$\begin{aligned} P_0(x,y,z) &= \bar{x} \vee \bar{y} \vee \bar{z}, \\ P_1(x,y,z) &= x \vee \bar{y} \vee \bar{z}, \\ P_2(x,y,z) &= x \vee y \vee \bar{z}, \\ P_3(x,y,z) &= x \vee y \vee z. \end{aligned}$$

1.3 Plan of this course

The course studies three main problems, from three different areas:

1. How difficult is deciding whether an instance of CSP($\mathcal{I}$) is satisfiable?
2. How well can we approximate the maximum number of satisfied constraints for an instance of CSP($\mathcal{I}$)?
3. Can we prove that unsatisfiable instances of SAT are indeed unsatisfiable?

Apart from these problems, we might also study several other subjects:

- The complexity of SAT under the promise that there is at most one solution.
- Counting and sampling solutions of kSAT.
- Exponential time algorithms for kSAT.
- Algorithms which certify that a random kSAT is unsatisfiable.

Let us now expand on the three main problems highlighted above.

Complexity of the satisfiability version of CSP($\mathcal{I}$) It is well-known that 3SAT is NP-complete, whereas 2SAT is in P. Are there other possibilities?

Ladner's theorem [Lad75], proved via diagonalization, states that if P≠NP then some language in NP is neither in P nor NP-hard; we call such problems *NP-intermediate*. Are there natural such problems? Candidates include decision versions of Factoring, the Graph Isomorphism problem, and meta-complexity problems such as the Minimum Circuit Size Problem (MCSP). A celebrated result of Schaefer, known as *Schaefer's dichotomy theorem* [Sch78], states that in the setting of Boolean CSPs, CSP($\mathcal{I}$) is never NP-intermediate, that is, for each finite $\mathcal{I}$, satisfiability of CSP($\mathcal{I}$) is either in P or it is NP-complete. Feder and Vardi [FV99] conjectured that a similar result holds for any finite domain. A long line of work has recently succeeded in proving this conjecture, known as the *dichotomy theorem* [Bul17, Zhu20].

We will present a modern proof of Schaefer's theorem, following notes of Hubie Chen [Che09], which uses *universal algebra*. In contrast to Schaefer's original proof, the modern proofs give an efficient algorithm for determining whether CSP($\mathcal{I}$) is easy or hard.

Complexity of the optimization version of CSP($\mathcal{I}$) Consider the problem 3XOR, in which each constraint is of the form $x_i \oplus x_j \oplus x_k = b$. A random assignment satisfies half of the constraints, and this gives a randomized 1/2-approximation to MAX-3XOR, the optimization version of 3XOR; the algorithm can be derandomized in various ways. Can we do better, in (randomized) polynomial time?

Let the *value* of a 3XOR instance be the maximum fraction of constraints satisfied by any satisfying assignment. Håstad [Hås01] showed that it is NP-hard to tell apart instances with value $\geq 1-\epsilon$ from instance with value $\leq 1/2+\epsilon$, for any constant $\epsilon > 0$. In other words, there is a polynomial time reduction f from SAT to MAX-3XOR such that if φ is satisfiable then $\mathrm{val}(f(\varphi)) \geq 1-\epsilon$, and if φ is unsatisfiable then $\mathrm{val}(f(\varphi)) \leq 1/2+\epsilon$. This shows that the trivial random assignment algorithm is in fact optimal!

Håstad's proof builds on the PCP theorem [AS98, ALM+98] and on Raz's parallel repetition theorem [Raz98], two difficult results which we will not cover in class. Håstad makes heavy use of *Boolean function analysis* [O'D14], in a relatively elementary form. We will cover all necessary preliminaries in class.

Another prominent problem is MAX-CUT. A random assignment cuts half the edges, and so gives a 1/2-approximation. Using semidefinite programming, Goemans and Williamson [GW95] gave a 0.878-approximation algorithm. The approximation ratio itself is the solution of some optimization problem. Surprisingly, this approximation ratio is best possible, assuming the Unique Games Conjecture [Kho02]. We will describe the algorithm and its analysis in class (the only part that we skip is an efficient algorithm for solving SDPs); the unique games conjecture; and the hardness proof that shows that 0.878 is the optimal approximation ratio [KKMO07].

Raghavendra [Rag08] generalized the work on MAX-CUT to arbitrary CSPs. He showed that every CSP has an optimal approximation algorithm (assuming the unique games conjecture) which uses semidefinite programming. We will not describe this very interesting result.

Proving that CSP($\mathcal{I}$) **is unsatisfiable** We can prove that a SAT instance is satisfiable by giving a satisfying assignment; this is the essence of NP. Can we similarly prove that a given instance is unsatisfiable? We don't expect this to be the case, since otherwise NP=coNP, but can we point out specific instances which are difficult to refute in particular ways?

SAT solvers are algorithms that get as input a SAT instance φ, and output either a satisfying assignment or a transcript of the algorithm, which constitutes a proof that φ is unsatisfiable. SAT solvers are based on a simple algorithm known as DPLL [DP60, DLL62]. At each point, the algorithm chooses a variable, and tries both assignments to it; a leaf which contradicts a clause of φ can be pruned away. Modern SAT solvers use an additional trick: if the current assignment is already known to cause a contradiction, then it can also be pruned away; this is known as *clause learning* [MSS99, BS97], and this type of algorithm is known as CDCL.

It turns out that when run on an unsatisfiable CNF, the transcript of DPLL solvers can be converted to a proof system known as *treelike Resolution*, and the transcript of CDCL solvers can be converted to a stronger proof system known as *(general) Resolution*. Using a technique of Ben-Sasson and Wigderson [BSW01], we

will describe simple unsatisfiable CNFs which are hard to refute in Resolution. Using the same technique, we will also show that random CNFs are hard to refute in Resolution.

1.4 Technical snippets

Let us conclude this introduction with a few technical snippets.

Deterministic 1/2-approximation algorithm for MAX-3XOR Above we mentioned a trivial 1/2-approximation algorithm for MAX-3XOR: choose a random assignment. This algorithm is easy to derandomize using the method of *conditional expectations*. The idea is to compute, for a given MAX-3XOR instance φ and partial assignment α, the expected number $E(\alpha)$ of satisfied constraints in a random truth assignment extending α. The calculation is easy, since each constraint is either satisfied by α, refuted by α, or is satisfied by a random extension of α with probability 1/2.

How does this help us? Let $x_1, \ldots, x_n$ be the variables appearing in the instance, and suppose that the instance has m constraints. We know that the expected number of constraints satisfied by a random assignment is $m/2$. By calculating $E(x_1 = 0)$ and $E(x_1 = 1)$, we can find an assignment α_1 to x_1 such that $E(x_1 = \alpha_1) \geq m/2$. Continuing in this way, we can find assignments for the other variables, maintaining the invariant the the current partial assignment α satisfies $E(\alpha) \geq m/2$. Eventually we obtain a total assignment which satisfies at least $m/2$ constraints.

When calculating $E(x_1 = \alpha_1)$, the only terms which depend on α_1 correspond to constraints of the form $x_1 = b$: each such constraint contributes 1 to $E(x_1 = b)$ and 0 to $E(x_1 = \bar{b})$. This gives another description of the algorithm: choose the value of x_i which maximizes the number of constraints of the form $x_i = b$ which are satisfied. In other words, this is just the greedy algorithm.

Exercise Generalize this algorithm to MAX-E3SAT, the variant of MAX-3SAT in which each clause consists of three *different* literals.

Finding a solution using a SAT oracle Using similar ideas, we can reduce the search version of SAT (given a satisfiable instance, find some satisfying assignment) to its decision version. Given a CNF φ and access to a SAT oracle, we proceed as follows. First, we find a value α_1 of x_1 such that $\varphi|_{x_1=\alpha_1}$ is satisfiable. We proceed in this way, finding values for the other variables, until we find a satisfying assignment.

Exercise #SAT is the problem of counting the number of solutions to a SAT instance. Show that using an oracle to #SAT you can sample a random satisfying assignment of a given CNF.

Exercise Show how to find a legal 3-coloring of a 3-colorable graph using an oracle for the decision version of 3-coloring.

2 Schaefer's theorem

This section mostly follows Hubie Chen's charming notes [Che09], with additional proofs from Víctor Dalmau's PhD thesis [Dal00].

2.1 Gadget reductions

SAT is the prototypical NP-complete problem. Its special case, 3SAT, is also NP-complete. Let us now show that two more variants are NP-complete: 1-in-3SAT and NAE-3SAT.

1-in-3SAT Let P be the following predicate on three Boolean variables:

$$P = \{(1,0,0),(0,1,0),(0,0,1)\}.$$

An instance of 1-in-3SAT is a collection of constraints of the form $P(x,y,z)$, where x,y,z are arbitrary *literals*. (Since we are allowing literals rather than variables, this is not quite CSP(P), but it is CSP($\mathcal{P}$) for a slightly larger collection of predicates.)

We can show that 1-in-3SAT is NP-hard using a gadget reduction from 3SAT, which translates a disjunction of three literals to an equivalent collection of 1-in-3 clauses. Let the disjunction in question be $x \vee y \vee z$, where each of x,y,z is a literal. Consider the following matrix of variables:

x	x'	x''
y	y'	y''
z	z'	z''

The constraints are:

- Each row contains exactly one satisfied literal.
- The second and third columns each contain *at most* one satisfied literal.

To express the *at most* constraint, say on the second column, we use the following gadget:

$$P(x',y',u') \wedge P(x',z',v') \wedge P(y',z',w').$$

This clearly implies that at most one of x',y',z' is satisfied, since if, for example, both x' and y' are satisfied, then $P(x',y',u')$ cannot hold. Conversely, if none of x',y',z' is satisfied, we can satisfy the constraints by taking u',v',w' to be true, and if only x' is satisfied, then we can take u',v' to be false and w' to be true.

In total, $x \vee y \vee z$ is expressed as

$$\begin{aligned} &P(x,x',x'') \wedge P(y,y',y'') \wedge P(z,z',z'') \\ \wedge &P(x',y',u') \wedge P(x',z',v') \wedge P(y',z',w') \\ \wedge &P(x'',y'',u'') \wedge P(x'',z'',v'') \wedge P(y'',z'',w''). \end{aligned}$$

Why does this work? A satisfying assignment corresponds to a choice of one satisfied literal per row, with each column being selected at most once. If there are at most two unsatisfied literals among x,y,z, then we can choose the columns accordingly. In contrast, if all of x,y,z are unsatisfied, then the pigeonhole principle shows that we much choose one of the other two columns twice.

We use different variables $x',y',u',v',w',x'',y'',u'',v'',w''$ for each clause. This is crucial in order to guarantee the soundness of the reduction.

NAE-4SAT and NAE-3SAT Let Q be the following predicate on four Boolean variables:

$$Q = \{0,1\}^4 \setminus \{(0,0,0,0),(1,1,1,1)\}.$$

An instance of NAE-4SAT is a collection of constraints of the form $Q(x,y,z,w)$, where x,y,z,w are arbitrary literals. Here NAE stands for *not all equal*, since $Q(x,y,z,w)$ holds if x,y,z,w do not all have the same truth value.

We can show that NAE-4SAT is NP-hard using a different type of reduction from 3SAT, which is more global. We pick a new variable v, and add it to all clauses. That is, we replace each constraint $x \vee y \vee z$ with the constraint $Q(x,y,z,v)$, where the variable v is common to all clauses.

Why does this work? In one direction, we can extend an assignment satisfying the original 3SAT instance to one satisfying the NAE-4SAT instance by simply setting v to false. In the other direction, we consider two cases, depending on the value of v in an assignment satisfying the NAE-4SAT instance. If v is false, then we simply erase it to obtain an assignment satisfying the 3SAT instance. If v is true, then we flip the truth values of all variables; we get another assignment satisfying the NAE-4SAT instance, and so can proceed as before.

Is there a gadget reduction like the one we had before? We will define such reductions formally later, but we can already highlight an issue here. If an assignment satisfies an NAE-4SAT instance, then its complement (obtained by flipping all truth values) also satisfies the instance. This means that any NAE-4SAT instance which excludes assignments in which all of x,y,z are false, will necessarily also exclude assignments in which all of x,y,z are true. In other words, since this symmetry is present in NAE-4SAT but absent from 3SAT, we cannot expect a gadget reduction from 3SAT to NAE-4SAT.

As an aside, it is possible to gadget-reduce NAE-4SAT to its width 3 analog:

$$\mathrm{NAE}(x,y,z,w) \longleftrightarrow \mathrm{NAE}(x,y,t) \wedge \mathrm{NAE}(z,w,\neg t).$$

This is just the analog of the usual reduction from SAT to 3SAT.

2.2 Arc consistency

The original proof of Schaefer's theorem focused on three types of CSPs which are efficiently solvable: linear equations, 2SAT, and Horn-SAT (a version of SAT in which each clause contains at most one positive literal). Instead of 2SAT and Horn-SAT, we will present, in this section and the following one, two general strategies for solving CSPs, and identify classes of CSPs in which these strategies work. These algorithm will point the way to the algebraic proof of Schaefer's theorem. The connection to the original proof is explored in Section 2.7.

Consider a Boolean CSP instance consisting of constraints $P_1,\ldots,P_m$, where P_j has arity d_j and is applied to the variables $x_{j,1},\ldots,x_{j,d_j}$. We will maintain a *domain* of possible values for each variable. Originally, the domain of each variable x_i is $D_i = \{0,1\}$. Suppose that for some constraint P_j and for some variable $x_{j,k}$ appearing in the constraint, the following happens:

> Every satisfying assignment of $P_j(x_{j,1},\ldots,x_{j,d_j})$ in which $x_{j,1} \in D_{j,1},\ldots,x_{j,d_j} \in D_{j,d_j}$ satisfies $x_{j,k} = b$.

In this case, we know that any assignment satisfying the CSP must have $x_{j,k} = b$, and accordingly, we can set $D_{j,k} = \{b\}$. Similarly, for some constraint P_j it might be the case that *no* satisfying assignment of $P_j(x_{j,1},\ldots,x_{j,d_j})$ satisfies $x_{j,1} \in D_{j,1},\ldots,x_{j,d_j} \in D_{j,d_j}$. If this is the case, then we can declare that the instance is unsatisfiable.

Each variable can be pruned at most once, so the pruning process stabilizes within n steps, with one of the following outcomes. Either the instance was found to be unsatisfiable, or the following holds:

> For each constraint P_j, for each variable $x_{j,k}$, and for each $b \in D_{j,k}$, there is a satisfying assignment of

$P_j(x_{j,1}, \ldots, x_{j,d_j})$ in which $x_{j,1} \in D_{j,1}, \ldots, x_{j,d_j} \in D_{j,d_j}$ and $x_{j,k} = b$.

In general, this doesn't guarantee that the instance is satisfiable. For example, consider the 2SAT instance

$$(x \vee y) \wedge (\neg x \vee y) \wedge (x \vee \neg y) \wedge (\neg x \vee \neg y).$$

Here $D_x = D_y = \{0, 1\}$, and no truth value can be pruned away, yet the instance is unsatisfiable. Nevertheless, in some cases, we are able to guarantee that the instance is satisfiable. It is natural to try to extract a satisfying assignment from the domains $D_1, \ldots, D_n$. There are two canonical choices: $b_i = \max(D_i)$ and $b_i = \min(D_i)$. Let us check when does the choice $b_i = \max(D_i)$ work.

Let us consider an arbitrary constraint $P_j(x_{j,1}, \ldots, x_{j,d_j})$. By definition, for each $k \in \{1, \ldots, d_j\}$ there exists an assignment $x^{(k)}_{j,1}, \ldots, x^{(k)}_{j,d_j}$ satisfying P_j such that $x^{(k)}_{j,k} = b_{j,k}$ and $x^{(\ell)}_{j,\ell} \in D_{j,\ell}$ for all $\ell \neq k$. If $b_{j,\ell} = 0$ then $D_{j,\ell} = \{0\}$ and so $x^{(k)}_{j,\ell} = 0$ for all k, and if $b_{j,\ell} = 1$ then $x^{(\ell)}_{j,\ell} = 1$. Therefore $b_{j,\ell} = x^{(1)}_{j,\ell} \vee \cdots \vee x^{(d_j)}_{j,\ell}$. In other words,

$b_{j,1}, \ldots, b_{j,d_j}$ is a disjunction of satisfying assignments of P_j.

We conclude that the algorithm works whenever each predicate P_j satisfies the following property: *the disjunction of satisfying assignments is also a satisfying assignment.* In this case, we say that the function $g(x, y) = x \vee y$ is a *polymorphism* of P_j.

Polymorphisms have already showed up in the preceding section: NAE-4SAT had negation $g(x) = \neg x$ as a polymorphism, and this prevented a gadget reduction from 3SAT to NAE-4SAT. This connection between polymorphisms and gadget reductions is the driving force behind the algebraic approach to Schaefer's theorem.

Exercise When does the choice $b_i = \min(D_i)$ work?

Exercise Extend the algorithm to the non-Boolean case. When does it work?

2.3 Extending partial solutions

Before elucidating the connection between polymorphisms and gadget reductions further, let us describe a generalization of arc consistency, in which we keep track of domains of sets of variables. We use the same setup as the preceding section: the constraints are $P_1, \ldots, P_m$, where P_j is a constraint on $x_{j,1}, \ldots, x_{j,d_j}$.

In the preceding section, we kept track of the domains of single variables. Now we want to keep track of the domains of small sets of variables, say of size ℓ (the size has to be constant in order to keep the algorithm efficient). For each set S of up to ℓ variables, we keep a domain $D_S \subseteq \{0, 1\}^S$, initialized at $\{0, 1\}^S$. We then prune these sets, as in the arc consistency algorithm: for each partial assignment $b_S \in D_S$ and each constraint P_j whose domain includes S, we check whether P_j has a satisfying assignment x in which $x_S = b_S$ and $x_T \in D_T$ for all $T \subseteq \mathrm{dom}(P_j)$ of size at most ℓ. If this is not the case, we remove b_S from D_S and continue.

Another reason to prune an assignment is even simpler: any $b_S \in D_S$ whose restriction to $T \subset S$ isn't contained in D_T can be removed. Similarly, any $b_T \in D_T$ which has no extension in D_S for some $S \supseteq T$ can be removed. After $O(n^\ell)$ pruning steps of both types, either one of the domains D_S shrinks to the empty set, in which case the instance is unsatisfiable; or otherwise, the domains D_S satisfy the following three properties:

- For each constraint P_j, for each $S \subseteq \mathrm{dom}(P_j)$ of size at most ℓ, and for each $b_S \in D_S$, there exists a satisfying assignment x of P_j such that $x_T \in D_T$ for all $T \subseteq \mathrm{dom}(P_j)$ and $x_S = b_S$.
- The restriction of each $b_S \in D_S$ to $T \subset S$ belongs to D_T.

- If $T \subseteq S$ and $b_T \in D_T$ then some $b_S \in D_S$ satisfies $b_S|_T = b_T$.

For each set S of size at most ℓ, each $b_S \in D_S$ is a *partial solution*, that is, an assignment of a subset of the variables which doesn't cause an immediate contradiction. Formally, for each constraint P_j, the restriction of b_S to the domain of P_j can be extended to a satisfying assignment of P_j. Indeed, if $T = S \cap \mathrm{dom}(P_j)$, then the restriction belongs to D_T, and so can be extended to a satisfying assignment.

We are given partial solutions for every set of variables of size ℓ. If we could extend it to larger and larger partial solutions, we would eventually obtain a partial solution for the set of all variables, which is just a satisfying assignment. How can such an extension process work out? Let us try to construct a partial solution for $x_1, \ldots, x_{\ell+1}$. For every $i \in \{1, \ldots, \ell+1\}$, if we remove x_i then there is some partial solution $b_1^{(i)}, \ldots, b_{i-1}^{(i)}, b_{i+1}^{(i)}, \ldots, b_{\ell+1}^{(i)}$ for the remaining variables. How do we combine them to a single assignment? In the preceding section, we used the OR (or AND) function. Another intuitive choice is the *majority* function, assuming for simplicity that ℓ is odd. Concretely, suppose that $\ell = 3$. We wish to construct a partial solution for x, y, z, w given the following partial solutions:

x	y	z	w
a_1	b_1	c_1	
a_2	b_2		d_2
a_3		c_3	d_3
	b_4	c_4	d_4

Consider some constraint P_j whose domain contains only x, y, z. It could be that a_1, b_1, c_1 is the only partial solution of P_j, while the majority vote produces a different assignment. In order to solve this problem, we strengthen our inductive goal.

The sets D_S that we start with satisfy one more property: if $|S| = \ell - 1$ and $b_S \in D_S$, then for each $v \notin S$ we can extend b_S to a partial solution of $S \cup \{v\}$ (just take any $b_{S\cup\{v\}} \in D_{S\cup\{v\}}$). We call this the $(\ell-1)$-*extension property*. We will attempt to show that for $r \geq \ell - 1$, the r-extension property implies the $(r+1)$-extension property. If this holds then we can start with an arbitrary partial solution of $x_1, \ldots, x_{\ell-1}$ and extend it to the variables $x_\ell, \ldots, x_n$, one by one, eventually constructing a satisfying assignment.

Let us try to prove the ℓ-extension property given the $(\ell-1)$-extension property. Suppose that we are given a partial solution $b_1, \ldots, b_\ell$, and we want to extend it to $x_{\ell+1}$. Just as before, it is natural to remove each b_i in turn, and use the $(\ell-1)$-extension property to get some value $b_{\ell+1}^{(i)}$ for $x_{\ell+1}$. Once again, it is natural to aggregate these values by taking a majority vote. Concretely, suppose that $\ell = 3$. The partial solutions we are considering, for variables x, y, z, w, are

x	y	z	w
a	b		d_1
a		c	d_2
	b	c	d_3

We let d be the majority of d_1, d_2, d_3. When is a, b, c, d a partial solution? It is clearly a partial solution for any constraint which doesn't involve w. Now consider any constraint P_j which does involve w, say one which involves all variables (as will transpire, this doesn't matter). We can "fill in the blanks" to obtain partial solutions of P_j which involve all variables:

x	y	z	w
a	b	c'	d_1
a	b'	c	d_2
a'	b	c	d_3

Observe that a, b, c, d is the *majority* of all of these partial solutions. In other words, if each constraint P_j is closed under the majority operation, then the 2-extension property implies the 3-extension property.

From now on, we consider only the case $\ell = 3$, and assume that all constraints P_j are invariant under the majority operation. A very similar proof can be used to show that the r-extension property implies the

$(r+1)$-extension property for all $r \geq 3$. Given a partial solution $b_1, \ldots, b_r$, instead of removing each b_i in turn, it suffices to remove b_1, b_2, b_3 in turn, and run exactly the same argument as above; the only difference is that the two tables contain more identical columns.

Using the terminology of the preceding section, we conclude that the algorithm works whenever each predicate P_j has $\mathrm{maj}(x, y, z)$ as a polymorphism.

Exercise Extend the algorithm to the non-Boolean case. What kind of polymorphism do the predicates P_j need in order for the algorithm to work, for a given value of ℓ?

2.4 Polymorphisms and invariants

The two algorithms just presented work whenever all predicates are invariant under some operation, which we called a polymorphism. Formally, a function $g\colon D^k \to D$ is a *polymorphism* of a predicate $R \subseteq D^m$ (which we think as a set of satisfying assignments) if the following holds for all arrays of assignments:

$$\begin{array}{ccc} b_1^{(1)} & \cdots & b_m^{(1)} \\ \vdots & \ddots & \vdots \\ b_1^{(k)} & \cdots & b_m^{(k)} \\ \hline g(b_1^{(1)}, \ldots, b_1^{(1)}) & \cdots & g(b_m^{(1)}, \ldots, b_m^{(k)}) \end{array}$$

If all the top rows are satisfying assignments of R, then so is the bottom row, which is obtained from the other rows by applying g to each column. As a shorthand, we can use $b^{(1)}, \ldots, b^{(k)}$ to represent the various rows, and then $g(b^{(1)}, \ldots, b^{(k)})$ represents the final row, obtained from the preceding rows by columnwise application of g.

Here are some examples:

- The constant function $g() = d$ is a polymorphism of R if $(d, \ldots, d)$ is a satisfying assignment.
- The projection functions $g(x_1, \ldots, x_k) = x_\ell$ are polymorphisms of all predicates.
- The majority function $\mathrm{maj}(x, y, z)$ is a polymorphism of $x_1 \vee x_2$: if $x_1 \vee x_2$, $y_1 \vee y_2$, $z_1 \vee z_2$ hold then by the pigeonhole principle, there exists $i \in \{1, 2\}$ such that at least two of x_i, y_i, z_i are true, which implies that their majority is true.
- The OR function $g(x, y) = x \vee y$ is a polymorphism of $x_1 \to x_2$: if both $x_1 \to x_2$ and $y_1 \to y_2$ hold then either one of x_2, y_2 is true, in which case $(x_1 \vee y_1, x_2 \vee y_2) = (?, T)$; or else both of x_2, y_2 are false, in which case both of x_1, y_1 must be false, hence $(x_1 \vee y_1, x_2 \vee y_2) = (F, F)$.
- The AND function $g(x, y) = x \wedge y$ is also a polymorphism of $x_1 \to x_2$: if both $x_1 \to x_2$ and $y_1 \to y_2$ hold then either one of x_1, y_1 is false, in which case $(x_1 \wedge y_1, x_2 \wedge y_2) = (F, ?)$; or else both of x_1, y_1 are true, in which case both of x_2, y_2 must be true, hence $(x_1 \wedge y_1, x_2 \wedge y_2) = (T, T)$.

If Γ is a set of predicates then we denote by $\mathrm{Pol}(\Gamma)$ the set of all functions which are polymorphisms of *all* predicates in Γ. The collection $\mathrm{Pol}(\Gamma)$ always contains all projections, and is closed under function composition. We call a collection satisfying these two properties a *clone*. The smallest clone containing a given family F of functions is called the *closure* of F, denoted $[F]$; the closure is obtained by adding all projections and closing the resulting set under composition.

Going in the other direction, a predicate R is an *invariant* of a function g if g is a polymorphism of R. If P is a set of functions then we denote by $\mathrm{Inv}(P)$ the set of all predicates which are invariant under all functions in P. What can we say about $\mathrm{Inv}(P)$? It respects *gadget reductions*.

Let us start by recalling the gadget reduction from the predicate $D(x,y,z) = x \vee y \vee z$ to the predicate $P(x,y,z)$, which is true when exactly one of x, y, z is true:

$$\begin{aligned} D(x,y,z) \longleftrightarrow \exists x', y', z', u', v', w', x'', y'', z'', u'', v'', w'' \\ P(x,x',x'') \wedge P(y,y',y'') \wedge P(z,z',z'') \\ \wedge P(x',y',u') \wedge P(x',z',v') \wedge P(y',z',w') \\ \wedge P(x'',y'',u'') \wedge P(x'',z'',v'') \wedge P(y'',z'',w''). \end{aligned}$$

This gives a *definition* of $D(x,y,z)$ in terms of the predicate P, which involves *existential quantification*. If P is invariant under any polymorphism g, then so is D. To see this, let us notate the situation more abstractly:

$$D(\vec{x}) \longleftrightarrow \exists \vec{\alpha} \phi_P(\vec{x}, \vec{\alpha}),$$

where ϕ_P is a conjunction of applications of P to various elements of $\vec{x}, \vec{\alpha}$; in our case, $\vec{x} = (x,y,z)$, $\vec{\alpha} = (x', \ldots, w'')$, and ϕ_P is the conjunction of 9 applications of P. We can think of $\vec{\alpha}$ as a *witness* for $\vec{x}$. Now suppose that g is a k-ary polymorphism of P. Given satisfying assignments $\vec{x}^{(1)}, \ldots, \vec{x}^{(k)}$, let $\vec{\alpha}^{(1)}, \ldots, \vec{\alpha}^{(k)}$ be corresponding witnesses. Then $g(\vec{\alpha}^{(1)}, \ldots, \vec{\alpha}^{(k)})$ witnesses that $g(\vec{x}^{(1)}, \ldots, \vec{x}^{(k)})$ is a satisfying assignment of D: all clauses of $\phi_P(\vec{x}, \vec{\alpha})$ are satisfied since g is a polymorphism of P.

Here is another example. Let $Q_+(x) = x$, $Q_-(x) = \bar{x}$, and $E(x,y) =$ "$x = y$"; the last one is known as the *equality predicate*. Consider two types of CSPs: $\mathrm{CSP}(Q_+, Q_-)$ and $\mathrm{CSP}(Q_+, Q_-, E)$. Both are trivial to solve (though the former is in coNLOGTIME while the latter is L-complete), but there is no gadget reduction from $\mathrm{CSP}(Q_+, Q_-, E)$ to $\mathrm{CSP}(Q_+, Q_-)$, since we can't express equality using monadic predicates. However, an algorithm for solving $\mathrm{CSP}(Q_+, Q_-)$ can be used to solve $\mathrm{CSP}(Q_+, Q_-, E)$, by eliminating the equality predicates. For each predicate $E(x,y)$, we simply eliminate y from the formula, replacing it with x. This suggests that we allow our gadget reductions to use the equality predicate for free. Indeed, just like projections are polymorphisms of all predicates, so the equality predicate is an invariant of all polymorphisms: if $x_1 = y_1, \ldots, x_k = y_k$ then $g(x_1, \ldots, x_k) = g(y_1, \ldots, y_k)$; this is the principle of substitution of equals for equals.

We are now ready to present the formal definition of gadget reduction. A predicate R is *pp-definable* from a set of predicates Γ if we can write

$$R(\vec{x}) \longleftrightarrow \exists \vec{\alpha}\, \phi(\vec{x}, \vec{\alpha}),$$

where ϕ is a conjunction of predicates from $\Gamma \cup \{E\}$ applied to arbitrary variables from $\vec{x}, \vec{\alpha}$. (Here *pp* is shortcut for *primitive positive*.)

If g is a polymorphism of Γ then it is a polymorphism of any relation which is pp-definable from Γ. In other words, $\mathrm{Inv}(P)$ is closed under pp-definability. Such a collection of predicates is known as a *coclone*. The smallest coclone containing a given set of predicates is called the *closure* of Γ, denoted $\langle \Gamma \rangle$; it is obtained by closing under pp-definitions.

2.5 Galois connection

In order to prove that $\mathrm{CSP}(\Delta)$ is NP-hard using gadget reductions, we start with a known NP-complete problem $\mathrm{CSP}(\Gamma)$, and show that each predicate Γ is pp-definable from Δ. In contrast, the algorithms in Sections 2.2 and 2.3 rely on polymorphisms of Δ. We would like to show that these two approaches are complementary: for each finite set of predicates Δ, either $\mathrm{Pol}(\Delta)$ is rich enough so that we can apply a known algorithm in order to solve $\mathrm{CSP}(\Delta)$, or we can reduce some NP-complete $\mathrm{CSP}(\Gamma)$ to $\mathrm{CSP}(\Delta)$.

If $\mathrm{Pol}(\Delta)$ is small then it has many invariants, hopefully enough to include some NP-complete CSP. We could show this by relating $\mathrm{Inv}(\mathrm{Pol}(\Delta))$ to $\langle \Delta \rangle$, the set of predicates pp-definable from Δ. One such relation is almost trivial: if R is pp-definable from Δ then R is invariant under all polymorphisms of Δ, as shown in the preceding section, and so $\langle \Delta \rangle \subseteq \mathrm{Inv}(\mathrm{Pol}(\Delta))$. Remarkably, the inverse inclusion also holds, whenever Δ is finite: every relation in $\mathrm{Inv}(\mathrm{Pol}(\Delta))$ is pp-definable from Δ !

Let R be any relation which is invariant under $\mathrm{Pol}(\Delta)$, and denote its satisfying assignments by $\vec{t}_1, \ldots, \vec{t}_m$. The set of satisfying assignments is closed under all m-ary polymorphisms of Δ. That is, if g is an m-ary polymorphism of Δ then $g(\vec{t}_1, \ldots, \vec{t}_m)$ also satisfies R. Conversely, if we choose g to be the projection π_i to the i'th coordinate, then we get $\pi_i(\vec{t}_1, \ldots, \vec{t}_m) = \vec{t}_i$. In other words, $\vec{t}$ is a satisfying assignment of R iff $\vec{t} = g(\vec{t}_1, \ldots, \vec{t}_m)$ for some m-ary polymorphism of Δ.

We can encode an m-ary polymorphism using D^m variables, whose names are $g(d_1, \ldots, d_m)$. The fact that g is an m-ary polymorphism of Δ translates to a conjunction of some predicates from Δ. Therefore we can define R as follows:

$$R(\vec{x}) \longleftrightarrow \exists g \text{ “}g \text{ is a polymorphism of } \Delta\text{”} \wedge \big(\vec{x} = g(\vec{t}_1, \ldots, \vec{t}_m)\big).$$

Let us illustrate this with $\Delta = \{Q_+, Q_-\}$, where $Q_+(x) = x$ and $Q_-(x) = \bar{x}$, and R being the equality relation $R(x, y) =$ “$x = y$”. The equality relation has two satisfying assignments, $\vec{t}_1 = (T, T)$ and $\vec{t}_2 = (F, F)$, and so $m = 2$. An m-ary function g is a polymorphism of Δ if

$$Q_+(g(T, T)) \wedge Q_-(g(F, F)).$$

We obtain the following definition of R:

$$R(x_1, x_2) \longleftrightarrow \exists g(T,T), g(T,F), g(F,T), g(F,F) \\ Q_+(g(T,T)) \wedge Q_-(g(F,F)) \wedge \big(x_1 = g(T,F)\big) \wedge \big(x_2 = g(T,F)\big).$$

Concluding, we have proved that $\mathrm{Inv}(\mathrm{Pol}(\Delta)) = \langle\Delta\rangle$ for all finite Δ.

It is natural to ask what happens if instead of starting with a set of predicates, we start with a set F of functions. If R is invariant under all functions in F, then each function in F is a polymorphism of R. In other words, $F \subseteq \mathrm{Pol}(\mathrm{Inv}(F))$. Since $\mathrm{Pol}(\cdot)$ contains all projections and is closed under composition, in fact $[F] \subseteq \mathrm{Pol}(\mathrm{Inv}(F))$. Does the converse hold? That is, do all polymorphisms of $\mathrm{Inv}(F)$ belong to $[F]$?

For every k, let $[F]_k$ consist of all k-ary functions in $[F]$. We can think of $[F]_k$ as a $|D|^k$-ary predicate whose variables encode the truth table of the k-ary function. Suppose that $g_1, \ldots, g_\ell \in [F]_k$ and $g \in [F]_\ell$. If we apply g on the truth tables of $g_1, \ldots, g_\ell$ then we obtain a k-ary function $g \circ (g_1, \ldots, g_\ell)$ given by

$$(g \circ (g_1, \ldots, g_\ell))(d_1, \ldots, d_k) = g(g_1(d_1, \ldots, d_k), \ldots, g_\ell(d_1, \ldots, d_k)).$$

This is just a composition of functions in F, and so $g \circ (g_1, \ldots, g_\ell) \in [F]_k$. In other words, $[F]_k \in \mathrm{Inv}(F)$. Now suppose that g is a k-ary polymorphism $\mathrm{Inv}(F)$. In particular, it is a k-ary polymorphism of $[F]_k$. In particular, if we apply g to the projection functions $\pi_1, \ldots, \pi_k$ (recall $\pi_i(d_1, \ldots, d_k) = d_i$), we should get another function in $[F]_k$. This application is

$$(g \circ (\pi_1, \ldots, \pi_k))(d_1, \ldots, d_k) = g(\pi_1(d_1, \ldots, d_k), \ldots, \pi_k(d_1, \ldots, d_k)) = g(d_1, \ldots, d_k).$$

In other words, it is g itself! Thus every k-ary polymorphism of $\mathrm{Inv}(F)$ belongs to $[F]_k$. Since k is arbitrary, this proves that $\mathrm{Pol}(\mathrm{Inv}(F)) = [F]$.

Summarizing, we have proved the following identities, the first requiring Δ to be finite:

- $\mathrm{Inv}(\mathrm{Pol}(\Delta)) = \langle\Delta\rangle$.
- $\mathrm{Pol}(\mathrm{Inv}(F)) = [F]$.

This shows that Pol and Inv are, in a sense, inverses.

We also have two monotonicity properties:

- If $\Gamma \subseteq \Delta$ then $\mathrm{Pol}(\Gamma) \supseteq \mathrm{Pol}(\Delta)$.
- If $F \subseteq G$ then $\mathrm{Inv}(F) \supseteq \mathrm{Inv}(G)$.

These properties should remind us of what happens in Galois theory, where Pol corresponds to the Galois group, and Inv corresponds to the fixed field. For this reason, this kind of connection is called a *Galois correspondence*.

2.6 Schaefer's theorem

Here is our general strategy. Given a finite set of predicates Γ, we consider $\mathrm{Pol}(\Gamma)$. If $\mathrm{Pol}(\Gamma)$ is small, then we want to conclude that $\mathrm{CSP}(\Gamma)$ is NP-hard; and if $\mathrm{Pol}(\Gamma)$ is large, then we want to use it to construct an efficient algorithm for $\mathrm{CSP}(\Gamma)$.

Suppose that Γ is a finite set of predicates such that $\mathrm{Pol}(\Gamma)$ is the minimal possible clone, consisting only of projections. Thus $\mathrm{Inv}(\mathrm{Pol}(\Gamma)) = [\Gamma]$ consists of all predicates. In particular, SAT is pp-definable in Γ, and so $\mathrm{CSP}(\Gamma)$ is NP-hard.

Another hard case which we have seen above is NAE-SAT. In contrast to SAT, NAE-SAT does have nontrivial polymorphisms: *anti-projections* $\pi_i(x) = \neg x_i$. If Γ is a finite set of predicates such that $\mathrm{Pol}(\Gamma)$ consists only of projections and anti-projections, then $\mathrm{Inv}(\mathrm{Pol}(\Gamma)) = [\Gamma]$ consists of all predicates which are invariant under negation. In particular, NAE-SAT is pp-definable in Γ, and so $\mathrm{CSP}(\Gamma)$ is again NP-hard.

Now suppose that we are given a finite set Γ of predicates which contain a polymorphism g that is neither a projection nor an anti-projection. We will now attempt to classify all such polymorphisms g, according to their arity. First, let us dispense with constant g. If $g(\vec{x}) = b$ is a polymorphism, then each predicate in Γ is either empty or satisfied by plugging in b for all variables. Such CSPs are easy to solve.

Now suppose that Γ has no constant polymorphisms, and let g be a polymorphism of minimal arity which is neither a projection nor an anti-projection. In particular, g depends on all coordinates. If g is binary, then it is one of the following:

$$\begin{array}{ll} x \vee y & x \wedge y \\ x \vee \bar{y} & x \wedge \bar{y} \\ \bar{x} \vee y & \bar{x} \wedge y \\ \bar{x} \vee \bar{y} & \bar{x} \wedge \bar{y} \\ x \oplus y & \overline{x \oplus y} \end{array}$$

In Section 2.2 we have described an algorithm, *arc consistency*, which can handle predicates invariant under $x \vee y$. Switching 0s and 1s, it can also handle predicates invariant under $x \wedge y$. Observe now that

$$\begin{array}{ll} x \vee \bar{x} = 1 & x \wedge \bar{x} = 0 \\ \bar{x} \vee x = 1 & \bar{x} \wedge x = 0 \\ \overline{\bar{x} \vee \bar{x}} \vee \overline{\bar{y} \vee \bar{y}} = x \vee y & \overline{\bar{x} \wedge \bar{x}} \wedge \overline{\bar{y} \wedge \bar{y}} = x \wedge y \\ x \oplus x = 0 & \overline{x \oplus x} = 1 \end{array}$$

Hence the remaining cases reduce to one of $0, 1, x \vee y, x \wedge y$.

In order to simplify the analysis of the remaining cases, let us show that we can further assume that $g(a, \ldots, a) = a$. Indeed, $h(a) = g(a, \ldots, a)$ is a polymorphism of Γ, and so by assumption, it is not constant. Hence either $h(a) = a$ or $h(a) = \neg a$. In the latter case, the function $G(\vec{a}) = g(\neg \vec{a})$ is a polymorphism of Γ of the same arity as g. Moreover, $G(a, \ldots, a) = g(\neg a, \ldots, \neg a) = a$. Finally, G cannot be a projection, since if $G(\vec{a}) = a_i$ then $g(\vec{a}) = \neg a_i$ would be an anti-projection. Therefore we can replace g with G in order to satisfy the constraint $g(a, \ldots, a) = a$.

Suppose now that g is ternary and depends on all coordinates. The function $g(a, a, b)$ is a polymorphism of Γ of smaller arity, and so g must be a projection or an anti-projection. Since $g(a, a, a) = a$, the function $g(a, a, b)$ must be a projection: either $g(a, a, b) = a$ or $g(a, a, b) = b$. Similarly, $g(a, b, a)$ and $g(b, a, a)$ are projections. Since any input of g contains two identical coordinates, the choices of projections completely determine g. There are eight different possibilities:

$g(a,a,b)$	a	a	a	a	b	b	b	b
$g(a,b,a)$	a	a	b	b	a	a	b	b
$g(b,a,a)$	a	b	a	b	a	b	a	b
	maj	π_1	π_2	ν_3	π_3	ν_2	ν_1	min

The first case is *majority*, handled by the algorithm of Section 2.3. The cases marked π_i are projections, and so cannot occur. The cases marked ν_i are majority with a single negated input. For example, $\nu_1(x, y, z) =$

$\mathrm{maj}(\bar{x}, y, z)$. To see this, we conside two cases. If $x = y = z$ then $\nu_1(x, y, z) = x = \mathrm{maj}(\bar{x}, y, z)$. Otherwise, we can write the input as an a, b pattern with $b = \bar{a}$. After flipping the first input, the majority is b, b, a, matching the column marked ν_1. We claim that

$$\mathrm{maj}(x, y, z) = \nu_1(\nu_1(x, y, z), y, z).$$

If $y = z$ then $\nu_1(*, y, z) = y = \mathrm{maj}(x, y, z)$. If $y \neq z$ then $\nu_1(x, y, z) = \bar{x}$, and so $\nu_1(\nu_1(x, y, z), y, z) = \nu_1(\bar{x}, y, z) = x = \mathrm{maj}(x, y, z)$.

The final case, also known as *minority*, is the operation $x \oplus y \oplus z$ (indeed, $a \oplus a \oplus b = b$ and so on). It turns out that in this case, each predicate $R \in \Gamma$ is either empty or an affine subspace, that is, a conjunction of linear equations over $GF(2)$; consequently, CSP(Γ) can be solved using Gaussian elimination. The proof is by induction on the arity of R.

If R is nullary then this is clear. Otherwise, let $R_0(\vec{y}) \leftrightarrow R(0, \vec{y})$ and $R_1(\vec{y}) \leftrightarrow R(1, \vec{y})$. By induction, each of R_0, R_1 is either empty or an affine subspace. If both of R_0, R_1 are empty then R is empty. If only R_0 is empty then R is the intersection of R_1 and the equation $x_1 = 1$. Similarly, if only R_1 is empty then R is the intersection of R_0 and the equation $x_1 = 0$.

Now suppose that both R_0 and R_1 are non-empty, choose $\vec{y}_0 \in R_0$ and $\vec{y}_1 \in R_1$, and let $\vec{z} = \vec{y}_0 \oplus \vec{y}_1$; thus $(0, \vec{y}_0), (1, \vec{y}_1) \in R$, and the XOR of these two is $(1, \vec{z})$. Since R is invariant under $a \oplus b \oplus c$,

$$\vec{y} \in R_0 \longrightarrow (0, \vec{y}) \in R \longrightarrow (1, \vec{y} \oplus \vec{z}) \in R \longrightarrow \vec{y} \oplus \vec{z} \in R_1,$$
$$\vec{y} \in R_1 \longrightarrow (1, \vec{y}) \in R \longrightarrow (0, \vec{y} \oplus \vec{z}) \in R \longrightarrow \vec{y} \oplus \vec{z} \in R_0.$$

In other words, if R_0 is the affine subspace $V + \vec{u}$, then R_1 is the affine subspace $V + \vec{u} + \vec{z}$. Therefore R is the affine subspace $V' + (0, \vec{u})$, where V' is the span of $(0, V)$ and $(1, \vec{z})$.

Finally, suppose that g has arity $m \geq 4$ and depends on all coordinates. We will show that this cannot happen, completing the proof of Schaefer's theorem.

As shown above, the function $g(a_1, a_1, a_3, a_3, a_5, \ldots, a_m)$ is a projection a_i. Without loss of generality, $i \neq 1$. The function $g(a_1, a_1, a_3, a_4, a_5, \ldots, a_m)$ is also a projection a_j. We cannot have $j = 1$, since if we identify a_3 and a_4, we get a_i for $i \neq 1$. Without loss of generality, suppose that $j = 4$. Identifying a_1 and a_3, we see that

$$g(a_1, a_1, a_1, a_4, a_5, \ldots, a_m) = a_4.$$

Now consider $g(a_1, a_2, a_2, a_4, a_5, \ldots, a_m)$. This is a projection a_k. If we identify a_1 and a_2, we get a_4, and so $k = 4$. Similarly, $g(a_1, a_2, a_1, a_4, a_5, \ldots, a_m) = a_4$. Thus

$$g(a, a, b, a_4, a_5, \ldots, a_m) = a_4,$$
$$g(a, b, a, a_4, a_5, \ldots, a_m) = a_4,$$
$$g(b, a, a, a_4, a_5, \ldots, a_m) = a_4.$$

Every input to g is captured by one of these three identities, and so $g(\vec{a}) = a_4$. That is, g is a projection, and so doesn't depend on all coordinates.

Finer classification Schaefer's theorem shows that every Boolean CSP is either NP-complete or in P. Allender, Bauland, Immerman, Schnoor, and Vollmer [ABI+09] refined this, by classifying the complexity of Boolean CSPs up to AC^0 reductions. They show that every Boolean CSP is either in coNLOGTIME, or it is hard with respect to AC^0 reductions for one of the following classes: L, NL, $\oplus$L, P, NP.

Post's lattice Post [Pos41] classified in 1941 all clones of Boolean functions. There are countably many cases, which can divided into finitely many families. The resulting poset of clones is a lattice known as *Post's lattice*. Post's classification can be used to replace the case analysis in the proof of Schaefer's theorem. Moreover, it was used by Allender et al. in their finer classification.

The situation for larger alphabets is wilder: there are uncountably many clones. Therefore the proof of the dichotomy theorem for general finite alphabets uses a case analysis similar to the current proof.

Polymorphisms of 3SAT and NAE-3SAT The proof of Schaefer's theorem shows that 3SAT and NAE-3SAT cannot have any polymorphisms other than projections and anti-projections, unless P=NP. We can in fact show it directly, using the completeness properties of these sets of perdicates.

It is well-known that every predicate is pp-definable in 3SAT. Explicitly, a predicate R is pp-definable in SAT as follows:

$$R(\vec{x}) \longleftrightarrow \bigwedge_{\vec{b} \notin R} \text{“}\vec{x} \neq \vec{b}\text{”}.$$

The predicate $\vec{x} \neq \vec{b}$ is just a disjunction of “$x_i \neq b_i$”. This can be further expressed in 3SAT using extension variables. The basic idea is to repeatedly use the identity

$$\alpha \vee \beta \longleftrightarrow \exists z\, (\alpha \vee z) \wedge (\beta \vee \neg z),$$

where α, β are arbitrary disjunctions. This allows us to express long clauses using ones of length 3. We can express shorter clauses using ones of length 3 by duplicating literals.

Since $\mathrm{Pol}(\mathrm{Inv}(\emptyset)) = [\emptyset]$ and $\mathrm{Inv}(\emptyset)$ consists of all relations, we conclude that the only polymorphisms of the set of *all* relations are projections. Since every relation is pp-definable in 3SAT, this implies that the only polymorphisms of 3SAT are projections.

NAE-3SAT does have polymorphisms, namely “anti-projections” $\pi_i(x) = \neg x_i$. We can show that the projections and the anti-projections are its only polymorphisms by mimicking the proof that we gave above for 3SAT. If R is a predicate which is invariant under complementation (which we denote by $\neg$) then

$$R(\vec{x}) \longleftrightarrow \bigwedge_{\vec{b} \notin R} \text{“}\vec{x} \neq \vec{b}, \neg\vec{b}\text{”}.$$

This shows that R is pp-definable in NAE-SAT. To express this in NAE-3SAT, we first observe that just as before, if α, β are sets of literals then

$$\mathrm{NAE}(\alpha, \beta) \longleftrightarrow \exists z\, \mathrm{NAE}(\alpha, z) \wedge \mathrm{NAE}(\beta, \neg z).$$

We leave the easy verification to the reader. This allows us to express long NAE-constraints as ones of width 3. We can express width 2 NAE as follows:

$$\text{“}x \neq y\text{”} \longleftrightarrow \exists z\, \mathrm{NAE}(x, y, z) \wedge \mathrm{NAE}(x, y, \neg z).$$

Since $\mathrm{Pol}(\mathrm{Inv}(\neg)) = [\neg]$, the set of all relations invariant under complementation has only projections and anti-projections as polymorphisms. Since every such relation is pp-definable in NAE-3SAT, we conclude that the only polymorphisms of NAE-3SAT are projections and anti-projections.

2.7 Alternative statement

The traditional account of Schaefer's theorem states that each Boolean CSP is either NP-hard or reduces to one of the following tractable cases:

1. The constant 0 assignment is always satisfying.
2. The constant 1 assignment is always satisfying.
3. 2SAT, in which each clause contains at most two literals.
4. Horn SAT, in which each clause contains at most one positive literal.
5. Dual-Horn SAT, in which each clause contains at most one negative literal.
6. Linear equations.

In our account, the tractable cases corresponded to having the following polymorphisms:

1. Constant 0.
2. Constant 1.
3. Majority.
4. Binary AND.
5. Binary OR.
6. Ternary XOR (Minority).

Items 1 and 2 in both lists clearly correspond, and our proof of tractability of Item 6 showed that it corresponds to linear equations. It turns out that the remaining items also correspond: a relation is invariant under Majority iff it is expressible as a 2CNF, and similarly for the other two cases. Since Horn SAT and Dual-Horn SAT are the same up to negative all variables, it suffices to discuss the cases of 2SAT and Horn SAT.

2SAT It is easy to check that 2SAT is invariant under majority. Let $\ell_1 \vee \ell_2$ be a 2SAT clause. Given three satisfying assignments, at least two of them satisfy the same literal ℓ_i; hence taking their majority will satisfy the same literal, and so the clause.

Conversely, suppose that R is a relation which is invariant under Majority. Let us show that it is expressible as a 2CNF (without extension variables!); this will imply that the polymorphisms of 2SAT are generated by Majority.

We will closely follow the algorithm for CSPs invariant under Majority that we presented in Section 2.3: we will show by induction that for $k \geq 2$, a $(k+1)$-ary relation invariant under Majority can be expressed as a kCNF. Given such a relation R, the strongest kCNF ϕ that we can choose consists of the conjunction of all k-clauses which are implied by R. For example, if no satisfying assignment $x = (x_1, \ldots, x_{k+1})$ of R has $x_1 = \cdots = x_k = 0$, then ϕ will include the clause $x_1 \vee \cdots \vee x_k$. In other words, ϕ states that the restriction of x to any k coordinates is a partial solution.

By construction, any satisfying assignment of R also satisfies ϕ. In the other direction, we use an argument identical to the analysis of the algorithm in Section 2.3. Let y be a truth assignment satisfying ϕ. Considering the projections to all variables but x_1, x_2, x_3 in turn, we deduce the existence of the following three satisfying assignments of R:

$$\begin{array}{cccc} * & y_2 & y_3 & y_4 \ldots \\ y_1 & * & y_3 & y_4 \ldots \\ y_1 & y_2 & * & y_4 \ldots \end{array}$$

The asterisks contain values which we have no control over. But whatever those values are, if we take the majority of all three satisfying assignments, we get back y.

How do we solve 2SAT efficiently? We can think of a 2SAT clause $\ell_1 \vee \ell_2$ as stating two statements:

- If ℓ_1 is false then ℓ_2 must be true.
- If ℓ_2 is false then ℓ_1 must be true.

Accordingly, we can construct a directed graph whose vertex set consists of all literals, and in which we include edges $\neg\ell_1 \to \ell_2$ and $\neg\ell_2 \to \ell_1$ for every clause $\ell_1 \vee \ell_2$. If this graph contains a directed path from a literal to its negation, then the 2SAT instance is clearly unsatisfiable. We will show that in any other case, the instance is satisfiable. The crucial observation is that if we have a path from ℓ_1 to ℓ_2 then we also have a path from $\neg\ell_2$ to $\neg\ell_1$.

Choose an arbitrary variable x. Either there is no directed path from x to $\bar{x}$, or there is no directed path from $\bar{x}$ to x (or both); suppose the former. We set x, and every literal reachable from x, to true, and remove

all falsified literals from the graph. For this to make sense, we need to know that not both y and $\bar{y}$ can be reached from x. Indeed, if there were paths from x to y and from x to $\bar{y}$ then there would be a path from y to $\bar{x}$, and so we would obtain a path from x to $\bar{x}$. Similarly, in the remaining graph there is no path from any literal ℓ to $\bar{x}$, since then there would be a path from x to $\neg\ell$, and so we would have set ℓ to false and would have consequently removed ℓ from the graph. Continuing in this fashion for all variables, we obtain a satisfying assignment.

We will see an alternative algorithm, due to Krom [Kro67], in Section 5.1.

Horn SAT Horn SAT, named after Alfred Horn, arises in logic programming. The Prolog clause

```
u :- p,q,r.
```

states that u is true if all of p, q, r are true, which translates to the Horn clause $u \vee \bar{p} \vee \bar{q} \vee \bar{r}$.[1]

Horn clauses are invariant under AND. Indeed, consider a Horn clause $(y_1 \wedge \cdots \wedge y_m) \to x$. Given two satisfying assignments, if both of them satisfy $y_1 \wedge \cdots \wedge y_m$, then both of them also satisfy x. Otherwise, their conjunction doesn't satisfy $y_1 \wedge \cdots \wedge y_m$.

Conversely, suppose that R is a relation which is invariant under AND. We will show that it is expressible as a Horn CNF (without extension variables!), which will imply that the polymorphisms of Horn SAT are generated by AND. Incidentally, since

$$(y_1 \wedge \cdots \wedge y_m) \to x \longleftrightarrow (y_1 \wedge z) \to x \ \wedge \ (y_2 \wedge \cdots \wedge y_m) \to z,$$

we conclude that the polymorphisms of Horn 3SAT are already generated by AND.

As in the case of 2SAT, we will show that R is equivalent to the strongest Horn CNF ϕ implied by R, that is, the conjunction of all Horn clauses implied by R. By construction, every satisfying assignment of R satisfies ϕ. In the other direction, consider any satisfying assignment y of ϕ, and suppose that it assigns true to the variables in the set S. For each $i \notin S$, the clause $x_S \to x_i$ does not appear in ϕ (where x_S signifies that all variables in S evaluate to true), and so there is a satisfying assignment of R which sets all variables in S to true, but sets x_i to false. Taking the AND of all these assignments for all $i \notin S$, we get y, and so y satisfies R.

How do we solve Horn SAT efficiently? Let us first observe that if there is no clause of the form x, then assigning false to all variables satisfies all clauses. Conversely, if there is a clause of the form x, then we must set x to true, and can then remove all occurrences of $\bar{x}$ from all remaining clauses. Continuing in this way, we either find a satisfying assignment, or arrive at a CNF in which one of the clauses is empty, concluding that the instance is unsatisfiable.

[1] In Gentzen's sequent calculus, this will be written $p, q, r \vdash u$. More generally, $P \vdash Q$ means that if *all* propositions in P hold then *at least one* of the propositions in Q holds.

3 Approximability: MAX-3LIN

This and the following section follow lecture notes of Prahladh Harsha (*Limits of approximation algorithms*, TIFR 2010) and of Ryan O'Donnell (*Advanced Approximation Algorithms*, CMU 18-854B, Spring 2008).

3.1 Approximation algorithms and inapproximability

Recall that in MAX-3XOR, also known as MAX-3LIN, we are given a bunch of linear equations on three variables $x_i \oplus x_j \oplus x_k = b$, and the goal is to find an assignment which satisfies as many equations as possible.

It is easy to check whether an instance is satisfiable using Gaussian elimination. What can we say when an instance is unsatisfiable? The *Razborov–Smolensky encoding* [Raz87, Smo87] relates satisfiability of clauses to satisfiability of linear equations. Given a 3-clause $x \vee y \vee z$, consider the set of linear equations of the form $\alpha x \oplus \beta y \oplus \gamma z = 1$, where α, β, γ go over all 7 possible values other than $(0, 0, 0)$. If $x = y = z = 0$ then all of these linear equations are false, and otherwise exactly 4 of them are true.

We can realize these linear equations in 3LIN by adding N copies of the equation $w \oplus w \oplus w = 0$, for large enough N, and replacing the constant 0 with w. This allows us to convert a 3CNF on m clauses ϕ to a MAX-3LIN instance ψ with $7m + N$ constraints such that ϕ is satisfiable iff some assignment satisfies at least $4m + N$ constraints of ψ (choosing $N = 3m + 1$ suffices). Thus the decision version of MAX-3LIN is NP-hard.

One standard way to cope with NP-hard problems is via *approximation algorithms*. In this case, we are looking for a polynomial time algorithm which, given an instance ψ of MAX-3LIN with *value* O (the value is the maximum fraction of constraints which can be simultaneously satisfied), produces a satisfying assignment which satisfies at least $c \cdot O$ of the constraints. The constant c is known as the *approximation ratio*. We often allow approximation algorithms to be randomized, and then we either measure the expected value of the assignment that they produce, or require them to output a good assignment with high probability.

A random assignment satisfies half the constraints in expectation; moreover, we can derandomize this algorithm to obtain a *deterministic* algorithm which always satisfies at least half the constraints. Since the value is always at most 1, this algorithm has approximation ratio 1/2. Can we do better?

If P=NP then we can solve MAX-3LIN exactly (this is a nice exercise). Following Håstad [Hås01], we will show that if P≠NP, then the approximation ratio of 1/2 cannot be improved upon. As in the usual theory of NP-completeness, we will aim at a many-one reduction from SAT to "approximating MAX-3LIN". Here is what this means in practice: for every $\epsilon > 0$, we will construct a polynomial time reduction that takes as input a SAT instance ϕ and outputs a MAX-3LIN instance ψ such that:

- **Completeness.** If ϕ is satisfiable then the value of ψ is at least $1 - \epsilon$.
- **Soundness.** If ϕ is not satisfiable then the value of ψ is at most $1/2 + \epsilon$.

In other words, the *promise version* of MAX-3LIN in which the instance is either at least $(1 - \epsilon)$-satisfiable or at most $(1/2 + \epsilon)$-satisfiable is NP-hard.

If we had an approximation algorithm for MAX-3LIN with approximation ratio $1/2 + \delta$ then we could use it to solve SAT as follows. First, we choose a value of ϵ such that

$$(1 - \epsilon)(1/2 + \delta) > 1/2 + \epsilon \iff \frac{1/2 + \epsilon}{1 - \epsilon} < 1/2 + \delta.$$

This is possible since as $\epsilon \to 0$, the ratio on the right-hand side tends to 1/2. Given a SAT instance ϕ, we reduce it to a MAX-3LIN instance ψ satisfying the properties above. We run the approximation algorithm, and calculate the value S of the assignment it produces. If $S > 1/2 + \epsilon$ then we known that ϕ must be satisfiable. Conversely, if ϕ is satisfiable then $S \geq (1 - \epsilon)(1/2 + \delta) > 1/2 + \epsilon$. Thus ϕ is satisfiable iff $S > 1/2 + \epsilon$.

Incidentally, this rules out approximation algorithms that only approximate the value of a MAX-3LIN instance, rather than produce an actual good assignment. We say that MAX-3LIN is NP-hard to approximate better than 1/2.

Perfect completeness The promise version of MAX-3LIN in which the instance is either satisfiable or at most $(1/2+\epsilon)$-satisfiable is easy: we can decide it using Gaussian elimination. The same cannot be said about MAX-3SAT.

We can relate the hardness of MAX-3SAT to that of MAX-3LIN by encoding a linear equation $x \oplus y \oplus z = 0$ as the corresponding CNF:

$$(x \vee y \vee \bar{z}) \wedge (x \vee \bar{y} \vee z) \wedge (\bar{x} \vee y \vee z) \wedge (\bar{x} \vee \bar{y} \vee \bar{z}).$$

If $x \oplus y \oplus z = 0$ then the CNF is satisfied, and otherwise exactly one clause is not satisfied. If we compose this encoding with the reduction from SAT to the promise version of MAX-3LIN, then we get a reduction from SAT to the promise version of MAX-3SAT with the following features, denoting the input SAT instance by ϕ and the output MAX-3SAT instance by χ:

- **Completeness.** If ϕ is satisfiable then the value of χ is at least $1 - \epsilon$.
- **Soundness.** If ϕ is not satisfiable then the value of χ is at most $(1/2+\epsilon)+(1/2-\epsilon)(3/4) = 7/8+\epsilon/4$.

This suffices to show that it is NP-hard to approximate MAX-3SAT better than $7/8$, which matches the trivial algorithm which chooses a random assignment, *if we assume that each clause mentions three different variables* (an assumption satisfied by the instances produced by our reduction, as it turns out).

We could ask for more: instead of the *imperfect* completeness appearing in this reduction, we could hope for *perfect* completeness, that is, for a reduction in which in the first case, χ is completely satisfiable. Such a reduction was found by Håstad [Hås01]. This shows that it is even hard to distinguish between satisfiable instances and those which are only marginally satisfiable beyond the random assignment threshold. We won't prove this result here.

Exercise Explain how to solve MAX-3LIN exactly using a SAT oracle.

3.2 PCP theorem, label cover, parallel repetition

PCP theorem The hardness results we are interested in concern promise problems in which there is a gap between Yes instances and No instances. The starting point of all such hardness proofs is the *PCP theorem*. There are many ways to state the PCP theorem. Let us start with the perspective of proofs.

Consider a game between two parties: Prover (she/her) and Verifier (he/him). Prover is trying to convince Verifier that she has a satisfying assignment for some 3CNF ϕ on n variables and m clauses, known to both parties. One obvious way to do so is to hand Verifier a satisfying assignment x for ϕ, which Verifier can check in polynomial time. What if we want Verifier to run even faster, say in constant time? He could choose a random clause C of ϕ, read the three corresponding variables of x, and check that they satisfy C.

If ϕ is satisfiable and Prover sends Verifier a satisfying assignment, then Verifier's check will always pass. When ϕ is not satisfiable, the best strategy for Prover is to choose an assignment which falsifies as few clauses as possible, say k. Verifier then catches Prover with probability k/m. The problem with this strategy is that k could be as small as 1, in which case Verifier only catches Prover with probability $1/m$.

The PCP theorem gives an efficiently computable encoding $E(\phi)$ of ϕ, which is another 3CNF satisfying:

- **Completeness.** If ϕ is satisfiable, then so is $E(\phi)$.
- **Soundness.** If ϕ is not satisfiable, then every assignment falsifies a γ-fraction of the clauses of $E(\phi)$, where $\gamma > 0$ is some absolute constant.

By running his previous strategy on $E(\phi)$, Verifier is able to catch Prover in the unsatisfiable case with probability $\gamma > 0$. He can magnify his success probability by running several rounds of this procedure (more on that, later).

Label Cover Another way to look at this game introduces another Prover, which holds a satisfying assignment for each clause. Let us call the two provers Clause-Prover and Variable-Prover. Verifier now chooses a clause C and a variable x_i contained in it; queries Clause-Prover for a satisfying assignment y of C, and Variable-Prover for the value z_i of x_i; and checks that $y_i = z_i$.

If ϕ is satisfiable, then Variable-Prover can choose a satisfying assignment z, and Clause-Prover can use the same assignment for her satisfying assignments. In contrast, suppose that any assignment falsifies at least k clauses of ϕ. Then whatever assignment z Variable-Prover chooses, there will be at least k clauses on which Clause-Prover will have to disagree with z, and so Verifier will catch the provers with probability at least $k/3m$.

A neat way to look at this new game is via a bipartite graph. The left side consists of the m clauses of ϕ, and the right side consists of the n variables of ϕ. We connect a clause C and a variable x_i if C mentions x_i. In the two-prover game, Clause-Prover chooses a 7-coloring of the left side, and Variable-Prover chooses a 2-coloring of the right side. Verifier then chooses a random edge and checks that the color $y(C)$ of C matches the color $z(x_i)$ of x_i. This check can be realized as a function $f\colon \{1,\dots,7\} \to \{0,1\}$, which we can conveniently write on the (C, x_i) edge.

The abstract version of this problem is known as *Label Cover*. For finite sets Σ, Δ, an instance of (Σ, Δ)-Label Cover consists of a bipartite graph $G = (V_L, V_R, E)$, and for each edge $e = (v_l, v_r) \in E$, a function $f_e\colon \Sigma \to \Delta$. The goal is to choose a Σ-coloring α of V_L and a Δ-coloring β of V_R such that $f_{v_l,v_r}(\alpha(v_l)) = \beta(v_r)$ for as many edges $(v_l, v_r) \in E$ as possible. The *value* of the instance is the maximum fraction of constraints that can be satisfied.

Using this language, we can restate the PCP theorem: Given a CNF ϕ, it constructs in polynomial time an instance Π of $(7,3)$-Label Cover such that:

- **Completeness.** If ϕ is satisfiable, then Π is satisfiable (has value 1).
- **Soundness.** If ϕ is not satisfiable, then the value of Π is at most $1-\delta$, for some $\delta > 0$.

In other words, given an instance of $(7,3)$-Label Cover, it is NP-hard to distinguish between the following two cases:

- Π is satisfiable.
- Π has value at most $1-\delta$.

All proofs of the PCP theorem are difficult, though the new proof of Dinur [Din07] is much shorter than the original proof [AS98, ALM+98].

Parallel repetition The gap provided by the PCP theorem, 1 versus $1-\delta$, is a good starting point, but for our actual hardness result we will need an arbitrarily large gap. The natural way to magnify the gap is via repetition: Verifier should ask Clause-Prover for several clauses, choose one variable each, and ask Variable-Prover for the corresponding variables. If Verifier employs t clauses, we would expect the value of the game to decrease as $(1-\delta)^t$. Surprisingly, this is not the case! The provers can sometimes do better.

Exercise Consider the following game, involving two provers and a verifier. Left Prover and Right Prover each choose a 2-coloring of the vertices $1,2,3$ of a triangle: Left chooses the colors $a_1, a_2, a_3 \in \{0,1\}$, and Right chooses the colors $b_1, b_2, b_3 \in \{0,1\}$. Verifier then chooses some $i \in \{1,2,3\}$, and either checks that $b_i = a_i$, or that $b_{i+1} \neq a_i$.

Model this game as an instance of Label Cover, and compute its value. Then consider what happens when Verifier asks each prover for the color of *two* vertices. Model the new game as an instance of Label Cover, and compute its value.

Raz's celebrated *parallel repetition theorem* [Raz98] states that if a game of this sort has value $1-\delta$, then repeating it enough times decreases the value arbitrarily close to zero. Quantitatively, Rao [Rao11]

(improving on the parameters obtained by Raz) showed that the value is at most $(1-\delta/2)^{\Omega(\delta t)}$ after t repetitions. The upshot is that for every $\epsilon > 0$ we can find alphabets Σ and Δ (whose size depends on ϵ) such that given an instance Π of (Σ, Δ)-label cover, it is NP-hard to distinguish between the following two cases:

- Π is satisfiable.
- Π has value at most ϵ.

This will be the starting point of our inapproximability result.

3.3 Long code, linearity testing (1)

Fix some small $\epsilon > 0$, and let Σ, Δ be alphabets such that it is NP-hard to distinguish satisfiable instances of (Σ, Δ)-Label Cover from instances whose value is at most ϵ. We would like to find a reduction that takes an instance Π of (Σ, Δ)-Label Cover and outputs an instance ψ of MAX-3LIN such that:

- **Completeness.** If Π is satisfiable then Ψ is almost satisfiable (has value close to 1).
- **Soundness.** If Π has value at most ϵ then Ψ has value roughly $1/2$.

The obvious idea is to have the variables of Ψ encode colorings of both sides of Π. How should we encode the color of a vertex v on the left-hand side? Let $\Sigma = \{1, \ldots, s\}$. We want to encode each $\sigma \in \Sigma$ using some collection $y_1, \ldots, y_m$ of bits. Each y_i is some function of σ. Which functions should we choose? There are only 2^s possible functions, which is a constant number, so why not just choose all of them? Here is a concrete example, with $s = 3$:

	000	001	010	011	100	101	110	111
1	0	0	0	0	1	1	1	1
2	0	0	1	1	0	0	1	1
3	0	1	0	1	0	1	0	1

Here the left column is an element $\sigma \in \{1, 2, 3\}$, and the remaining columns consist of all possible functions. We have suggestively provided indexes on the first row.

Generalizing from this special case, we are encoding each $\sigma \in \Sigma$ using a function $f_\sigma \colon \{0,1\}^s \to \{0,1\}$ given by $f_\sigma(x_1, \ldots, x_s) = x_\sigma$. In other words, f_σ is a *projection*, though in this setting, it usually goes by the name *dictator* (betraying the origins of Boolean function analysis in social choice theory). This encoding is known as the *Long Code*.

Suppose that our reduction encoded the color of a vertex v on the left-hand side by a function $f_v \colon \{0,1\}^{|\Sigma|} \to \{0,1\}$, whose intended value is f_σ, where $\sigma \in \Sigma$ is the color of v. Our reduction will need to test (implicitly or explicitly) that f_v is indeed a dictator, using the only means available to it: 3LIN constraints. How does such a test proceed?

Let us start with an easier goal: testing that a given function $f \colon \{0,1\}^n \to \{0,1\}$ is *linear*, that is, of the form $f(x) = \bigoplus_{i \in S} x_i$. A dictator is a special case of a linear function, so this seems like a good starting point. Another definition of *linear*, familiar from linear algebra, is: $f(x \oplus y) = f(x) \oplus f(y)$. The two definitions coincide, and the latter suggests a natural tester:

Choose $x, y \in \{0,1\}^n$ at random, and verify that $f(x \oplus y) = f(x) \oplus f(y)$.

If f is linear, then the test always passes; this is completeness. What about soundness? We cannot say much if f is close to a linear function. Indeed, if f is ϵ-close to a linear function, meaning that it results from a linear function by changing an ϵ-fraction of the entries, then the test will pass with probability at least $1 - 3\epsilon$. Therefore the most we can expect to say is that if the test passes with probability $1 - \epsilon$, then f is $O(\epsilon)$-close to a linear function.

It turns out that soundness does hold. There are at least three different proofs of this fact, and we will see two of them. The first one, which is also the original one, is quite intuitive; it is due to Blum, Luby and Rubinfeld [BLR93]. The second one uses Fourier analysis, which we cover later on; it is due to Bellare, Coppersmith, Haståd, Kiwi and Sudan [BCH$^+$96]. A third one, using induction, is due to David, Dinur, Goldenberg, Kindler and Shinkar [DDG$^+$17], who used their analysis to generalize the test to the case in which the function gets as input a binary vector of some constant even weight.

Let us therefore assume that $f(x \oplus y) = f(x) \oplus f(y)$ holds with probability $1-\epsilon$, where $\epsilon < 2/9$. We think of f as some noisy version of a genuine linear function g. How do we extract g from f? Writing the test as $f(x) = f(y) \oplus f(x \oplus y)$ suggests such a way: taking $g(x)$ to be the majority value of $f(y) \oplus f(x \oplus y)$. The resulting function is close to f: if $g(x) \neq f(x)$ then this means that the probability that $f(x) \neq f(y) \oplus f(x \oplus y)$ is at least $1/2$, implying that

$$\Pr[f(x) \neq f(y) \oplus f(x \oplus y)] \geq \frac{1}{2} \Pr[f(x) \neq g(x)].$$

Thus $\Pr[f \neq g] \leq 2\epsilon$.

Miraculously, the function g is linear! In order to show this, we first show that for every x, the value of $f(y) \oplus f(x \oplus y)$ is significantly skewed towards $g(x)$: namely,

$$\Pr[f(y) \oplus f(x \oplus y) = g(x)] > 2/3.$$

The idea is to relate this probability to the probability that $f(y) \oplus f(x \oplus y) = f(z) \oplus f(x \oplus z)$. If the former is $p > 1/2$ then the latter is $p^2 + (1-p)^2$, which is increasing with p. Therefore it suffices to show that $f(y) \oplus f(x \oplus y) = f(z) \oplus f(x \oplus z)$ with probability at least $5/9$.

Moving things around, it suffices to bound the probability that $f(y) \oplus f(x \oplus z) = f(z) \oplus f(x \oplus y)$. For every x, the values $y, x \oplus z$ are uniformly random, and so they equal $f(x \oplus y \oplus z)$ with probability more than $1 - 2/9$. The same can be said about the right-hand side, and so

$$\Pr[f(y) = f(x \oplus z) = f(x \oplus y \oplus z) = f(z) \oplus f(x \oplus y)] > 1 - 2/9 - 2/9 = 5/9.$$

Now we can show that g is linear. The idea is very simple: given x, y, each of the following holds with probability more than $2/3$ over the choice of z:

$$\begin{aligned} g(x) &= f(z) \oplus f(x \oplus z), \\ g(y) &= f(z) \oplus f(y \oplus z), \\ g(x \oplus y) &= f(x \oplus z) \oplus f(y \oplus z). \end{aligned}$$

Hence we can find a value of z for which all of these equations hold. Summing them, we see that $g(x \oplus y) = g(x) \oplus g(y)$. Since x, y were arbitrary, we conclude that g is linear, completing the analysis.

3.4 Fourier analysis and linearity testing (2)

The analysis of linearity testing that we just saw is very neat, but not as versatile as the next technique that we demonstrate: Boolean function analysis, which is the same as Fourier analysis on the group $\mathbb{Z}_2^n$.

We want to show that if $f(x \oplus y) = f(x) \oplus f(y)$ happens with high probability, then f is close to a linear function. It will be convenient to switch to a different representation of Boolean values, $\{-1, 1\}$, since XOR corresponds to multiplication in this representation (if we represent 0 as 1 and 1 as -1). Thus $f \colon \{\pm 1\}^n \to \{\pm 1\}$ satisfies $f(xy)f(x)f(y) = 1$ with high probability.

It is natural to express an arbitrary f as a *mixture* of linear functions, so we need names for these. For each S, we have a linear function $\chi_S(x) = \prod_{i \in S} x_i$, and in this context these are known as *Fourier characters*. The fundamental observation is that the Fourier characters form an orthonormal basis for the space of all functions on $\{\pm 1\}^n$ with respect to the inner product

$$\langle f, g \rangle = \mathop{\mathbb{E}}_x [f(x)g(x)] = \frac{1}{2^n} \sum_{x \in \{\pm 1\}^n} f(x)g(x).$$

Since $x_S^2 = 1$, the Fourier characters have unit norm. To see that they are orthogonal, suppose that $S \neq T$, and choose an arbitrary $i \in S \Delta T$. Then

$$\langle x_S, x_T \rangle = \frac{1}{2^n} \sum_{x \in \{\pm 1\}^n} \prod_{j \in S} x_j \prod_{k \in T} x_k = \frac{1}{2^n} \sum_{x \in \{\pm 1\}^n} \prod_{\ell \in S \Delta T} x_\ell = \frac{1}{2^n} \sum_{x_i \in \{\pm 1\}} x_i \sum_{x_{-i} \in \{\pm 1\}^{n-1}} \prod_{\substack{\ell \in S \Delta T \\ \ell \neq i}} x_\ell = 0,$$

where x_{-i} contains all variables but x_i. Finally, the Fourier characters form a basis since there are 2^n of them, matching the dimension of the space of all functions.

Since the Fourier characters form a basis, every function $f \colon \{\pm 1\}^n \to \mathbb{R}$ (not necessarily Boolean) has a unique expansion as a linear combination of Fourier characters, known as its *Fourier expansion*:

$$f(x) = \sum_{S \subseteq [n]} \hat{f}(S) \chi_S(x).$$

The coefficients $\hat{f}(S)$ are called *Fourier coefficients*.

What does all of this have to do with linearity testing? First, let us notice that if $\Pr[f(x)f(y)f(xy) = 1] = 1 - \epsilon$ then

$$\mathop{\mathbb{E}}_{x,y}[f(x)f(y)f(xy)] = (1 - \epsilon) - \epsilon = 1 - 2\epsilon.$$

The idea now is to open each of the copies of f into its Fourier expansion:

$$1 - 2\epsilon = \mathop{\mathbb{E}}_{x,y}[f(x)f(y)f(xy)] = \mathop{\mathbb{E}}_{x,y} \sum_S \hat{f}(S)\chi_S(x) \sum_T \hat{f}(T)\chi_T(y) \sum_U \hat{f}(U)\chi_U(xy).$$

Note that $\chi_U(xy) = \chi_U(x)\chi_U(y)$. Rearranging gives

$$1 - 2\epsilon = \sum_{S,T,U} \hat{f}(S)\hat{f}(T)\hat{f}(U) \mathop{\mathbb{E}}_{x,y}[\chi_S(x)\chi_T(y)\chi_U(x)\chi_U(y)] = \sum_{S,T,U} \hat{f}(S)\hat{f}(T)\hat{f}(U) \mathop{\mathbb{E}}_{x}[\chi_S(x)\chi_U(x)] \mathop{\mathbb{E}}_{y}[\chi_T(y)\chi_U(y)] = \sum_S \hat{f}(S)^3,$$

using orthonormality at the very last step. Orthonormality also implies that $1 = \mathbb{E}[f^2] = \sum_S \hat{f}(S)^2$, known in this context as *Parseval's identity*, and so

$$1 - 2\epsilon = \sum_S \hat{f}(S)^3 \leq \sum_S \hat{f}(S)^2 \cdot \max_T \hat{f}(T) = \max_T \hat{f}(T).$$

In other words, some T satisfies $\hat{f}(T) \geq 1 - 2\epsilon$. Applying orthonormality one last time, we see that

$$\hat{f}(T) = \langle f, \chi_T \rangle = \Pr[f = \chi_T] - \Pr[f \neq \chi_T] = 2\Pr[f = \chi_T] - 1.$$

We conclude that $\Pr[f = \chi_T] \geq 1 - \epsilon$. In other words, f is ϵ-close to the linear function χ_T.

3.5 Dictatorship testing and folding

Emboldened by our success with linearity testing, let us move on to our actual goal: dictatorship testing. Given a function $f \colon \{\pm 1\}^n \to \{\pm 1\}$, we want to test whether f is a dictator, that is, of the form $f(x) = x_i$ for some $i \in \{1, \ldots, n\}$.

How do we distinguish dictators from other linear functions? The basic idea is that if we slightly perturb the input, then a dictator would hardly be affected, whereas a function like parity (the linear function depending on all coordinates) will essentially reduce to a random function.

Given an input $x \in \{\pm 1\}^n$, we can perturb it into an input $y \in \{\pm 1\}^n$ in the following way: $y_i = x_1$ with probability $1 - \delta$, and $y_i = -x_i$ with probability δ. The effect of perturbation on Fourier characters is

easy to calculate. If we choose x uniformly at random and perturb it to y by negating each coordinate with probability δ, then

$$\Pr_{x,y}[\chi_S(x) = \chi_S(y)] = \frac{1}{2} + \frac{1}{2}\mathop{\mathbb{E}}_{x,y}[\chi_S(x)\chi_S(y)] = \frac{1}{2} + \frac{1}{2}\prod_{i\in S}\mathop{\mathbb{E}}_{x_i,y_i}[x_i y_i] = \frac{1}{2} + \frac{1}{2}(1-2\delta)^{|S|}.$$

Thus dictators are unaffected with probability $1-\delta$, whereas parity becomes almost completely random.

We can combine this idea with linearity testing by perturbing one of the three values x, y, xy, say the last one. One way to accomplish this is as follows: we choose x, y uniformly at random, let $z_i = 1$ with probability $1-\delta$ and $z_i = -1$ with probability δ, and check that $f(xyz)f(x)f(y) = 1$. This is known as the *dictatorship test* or *long code test*.

Dictators no longer pass this test with probability 1: the probability is reduced to $1-\delta$. That is, we lose perfect completeness. What about soundness? What can we say if $\Pr[f(x)f(y)f(xyz) = 1] \geq 1-\epsilon$? As before, this implies that

$$\begin{aligned}
1 - 2\epsilon &= \mathop{\mathbb{E}}_{x,y,z}[f(x)f(y)f(xyz)] \\
&= \sum_{S,T,U}\hat{f}(S)\hat{f}(T)\hat{f}(U)\mathop{\mathbb{E}}_{x,y,z}[\chi_S(x)\chi_T(y)\chi_U(xyz)] \\
&= \sum_{S,T,U}\hat{f}(S)\hat{f}(T)\hat{f}(U)\mathop{\mathbb{E}}_{x}[\chi_S(x)\chi_U(x)]\mathop{\mathbb{E}}_{y}[\chi_T(y)\chi_U(y)]\mathop{\mathbb{E}}_{z}[\chi_U(z)] \\
&= \sum_{S}\hat{f}(S)^3(1-2\delta)^{|S|}.
\end{aligned}$$

As before, this implies that

$$1 - 2\epsilon \leq \max_S \hat{f}(S)(1-2\delta)^{|S|}.$$

In particular, $\hat{f}(S) \geq 1-2\epsilon$, and so as before, $\Pr[f = \chi_S] \geq 1-\epsilon$. But we can say more: we also have

$$(1-2\delta)^{|S|} \geq 1 - 2\epsilon,$$

and so S is small.

Looking ahead, in our reduction to MAX-3LIN, the No case will involve instances whose value is close to $1/2$. This suggests that when analyzing the soundness of the dictator test, we should check what happens when the test succeeds with probability $1/2 + \eta$. Substituting $\epsilon := 1/2 - \eta$ gives us a set S satisfying the following properties:

- $\hat{f}(S) = \mathbb{E}[f\chi_S] \geq 1 - 2\epsilon = 2\eta$.
- $\Pr[f = \chi_S] \geq 1 - \epsilon = 1/2 + \eta$.
- $(1-2\delta)^{|S|} \geq 1 - 2\epsilon = 2\eta$, and so

$$|S| \leq \frac{\log(2\eta)}{\log(1-2\delta)} = O\left(\frac{1}{\delta}\log\frac{1}{\eta}\right).$$

How should we think of this kind of promise? It allows us to decode any function f which passes the dictatorship test with probability better than $1/2$ into a small set S of variables. This is somewhat similar to the concept of *list decoding*. In our context, the function f is supposed to encode a color $\sigma \in \Sigma$ of some vertex. We can think of S as a small list of such assignments which somehow arises from f. We will see below how such lists are useful. Roughly speaking, we will choose a color $\sigma \in S$ at random, and show that when doing so for all vertices in parallel, in expectation this produces a non-trivial coloring.

Folding Before proceeding any further, we should take note of one troubling phenomenon: while dictators don't pass the dictator test with probability 1, another function does, namely the constant 1 function, which corresponds to $\chi_\emptyset$. This is a problem since our decoding procedure would produce $S = \emptyset$ as the small list of suggestions for coloring the vertex encoded by f. We would like to rule out empty S.

One way to rule out the constant 1 function is to force the function f to be odd, that is to satisfy $f(-x) = -f(x)$ for all x, using *folding*. This is helpful, since dictators are odd whereas the constant function isn't. The way to enforce this constraint is very simple: we only specify the values of f on inputs in which $x_1 = 1$, and define f on the rest of the inputs using $f(x) = -f(-x)$ (the choice of x_1 is arbitrary).

We can analyze the test much as before, by treating f as an arbitrary odd function. Dictators still pass the test with probability $1 - \delta$. In the soundness analysis, we can say something more about the set S: it contains an odd number of elements, and so in particular, is not empty. Indeed, if f is an odd function and S contains an even number of elements then $\chi_S(-x) = (-1)^{|S|}\chi_S(x) = \chi_S(x)$, and so

$$\hat{f}(S) = \mathop{\mathbb{E}}_x[f(x)\chi_S(x)] = \mathop{\mathbb{E}}_x[f(-x)\chi_S(-x)] = -\mathop{\mathbb{E}}_x[f(x)\chi_S(x)] = -\hat{f}(S).$$

This implies that $\hat{f}(S) = 0$, and in particular it is not the case that $\hat{f}(S) \geq 2\eta$.

3.6 Hardness for MAX-3LIN

We are now in a position to construct a reduction from Label Cover to MAX-3LIN. Our starting point is an instance $\Pi = (U, V, E, \{\pi_e\})$ of (Σ, Δ)-Label Cover which is either satisfiable or at most γ-satisfiable, that is, at most a γ-fraction of constraints can be simultaneously satisfied. Here $\pi_e \colon \Sigma \to \Delta$ are the constraints (renamed from f_e to avoid collision with the long code encoding), and γ is a parameter of our choosing which determines the sizes of Σ and Δ. Our goal is to output an instance Ψ of MAX-3LIN with the following properties:

- **Completeness.** If Π is satisfiable then Ψ has value at least $1 - \epsilon$.
- **Soundness.** If Π is at most γ-satisfiable then Ψ has value at most $1/2 + \epsilon$.

Here $\epsilon > 0$ is a parameter given to us, and we can choose γ depending on ϵ.

Following the suggestions in the preceding sections, we will encode the color of each vertex in U by an odd function $f_u \colon \{\pm 1\}^\Sigma \to \{\pm 1\}$, which we store as a collection of $2^{|\Sigma|-1}$ Boolean variables. Similarly, we encode the color of each vertex in V by an odd function $f_v \colon \{\pm 1\}^\Delta \to \{\pm 1\}$, stored as a collection of $2^{|\Delta|-1}$ Boolean variables.

The intention is for each f_u and f_v to be a dictator. As we saw above, this can be enforced, to some extent, by checking that $f(x)f(y)f(xyz) = 1$ for each f, random x, y, and random z chosen so that $\Pr[z_i = 1] = 1 - \delta$ for each i. Incidentally, dictators pass this test only with probability $1 - \delta$, which is why the MAX-3LIN instance Ψ will not be satisfiable even when Π is.

Apart from checking that f_u and f_v are dictators for every u, v, we also want to check that the constraints π_e hold. Our strategy will be to choose an edge (u, v) at random, and check somehow that the dictators f_u, f_v agree with π_{uv}.

Suppose that f_u, f_v were indeed dictators satisfying π_{uv}. Then $f_u(x) = x_\sigma$ for some $\sigma \in \Sigma$, $f_v(y) = y_\tau$ for some $\tau \in \Delta$, and $\pi_{uv}(\sigma) = \delta$. We can therefore express f_v using f_u and π_{uv} as follows, where we assume that $\Sigma = \{1, \ldots, s\}$ and $\Delta = \{1, \ldots, t\}$:

$$f_v(y_1, \ldots, y_t) = f_u(y_{\pi_{uv}(1)}, \ldots, y_{\pi_{uv}(s)}).$$

Indeed,

$$f_u(y_{\pi_{uv}(1)}, \ldots, y_{\pi_{uv}(s)}) = y_{\pi_{uv}(\sigma)} = y_\tau = f_v(y).$$

Denoting the input to the left-hand side by $y \circ \pi_{uv}$, we can write this succinctly as $f_u(y \circ \pi_{uv}) = f_v(y)$. This gives us a way to convert vectors in $\{\pm 1\}^\Delta$ to corresponding vectors in $\{\pm 1\}^\Sigma$.

Our test will start by querying $f_u(x)$ at a random $x \in \{\pm 1\}^\Sigma$ and $f_v(y)$ at a random $y \in \{\pm 1\}^\Delta$. The only sensible thing to do now is to convert y into a vector in $\{\pm 1\}^\Sigma$ by composing with π_{uv}. Now we have two vectors $x, y \circ \pi_{uv} \in \{\pm 1\}^\Sigma$, and so we are led to the following test:

- Choose a random edge $(u, v) \in E$.
- Choose $x \in \{\pm 1\}^\Sigma$ and $y \in \{\pm 1\}^\Delta$ at random.
- Choose $z \in \{\pm 1\}^\Sigma$ at random in a biased manner: $z_i = 1$ with probability $1 - \delta$.
- Check whether $f_u(x) f_v(y) f_u(x(y \circ \pi_{uv})z) = 1$.

Our check, translated back to $\{0, 1\}$ variables, is of the form $p \oplus q \oplus r = b$, where p, q, r are Boolean variables (part of the encodings of f_u, f_v), and b is a constant. Although in the test as stated b is always equal to zero, when we undo the folding we get arbitrary b.

Our test is a probability distribution over equations of the form $p \oplus q \oplus r = b$. This is not quite an instance of MAX-3LIN: an instance is a *list* of equations rather than a *distribution*. However, this is not a big difference. Assuming that δ is a rational constant (depending on ϵ), we can find an integer N, whose size is polynomial in that of Π, such that the probability that each equation $p \oplus q \oplus r = b$ gets chosen is of the form M/N. We can express this by repeating the equation M times. The fraction of constraints satisfied by some truth assignment is then the same as the probability that the assignment satisfies a random constraint. (If we want to reduce the size of the instance, we can round M/N into a rational with smaller denominator; this only affects the results by a little bit.)

Analysis of the reduction It remains to prove that our reduction works. One direction, completeness, is easy: suppose that Π is satisfiable, say given by the coloring c. We set $f_u(x) = x_{c(u)}$ and $f_v(y) = y_{c(v)}$. We can do this since dictators are odd functions. Then

$$f_u(x) f_v(y) f_u(x(y \circ \pi_{uv})z) = x_{c(u)} \cdot y_{c(v)} \cdot x_{c(u)} y_{\pi_{uv}(c(u))} z_{c(u)} = z_{c(u)},$$

since $\pi_{uv}(c(u)) = c(v)$. Hence the test passes with probability $1 - \delta$. This means that completeness holds as long as we choose $\delta \leq \epsilon$.

The difficult direction is proving soundness. Here we follow the analysis of the dictatorship test, and see where it leads us. We will assume that

$$\Pr_{\substack{(u,v) \in E \\ x,y,z}} [f_u(x) f_v(y) f_u(x(y \circ \pi_{uv})z) = 1] > \frac{1}{2} + \epsilon,$$

and will attempt to deduce that the value of the Label Cover instance Π is more than γ.

Our assumption states that on average, the modified dictatorship test passes with probability more than $1/2 + \epsilon$. Hence on a typical edge, it passes with probability more than $1/2 + \epsilon$. Let us say that an edge is *good* if the test passes with probability at least $1/2 + \epsilon/2$. At least an $\epsilon/2$-fraction of edges are good, since otherwise the average success probability of the test would be smaller than

$$\frac{\epsilon}{2} \cdot 1 + \left(1 - \frac{\epsilon}{2}\right)\left(\frac{1}{2} + \frac{\epsilon}{2}\right) < \frac{\epsilon}{2} + \left(\frac{1}{2} + \frac{\epsilon}{2}\right) = \frac{1}{2} + \epsilon.$$

(This sort of calculation is known as an *averaging argument*.) From now on, when constructing a satisfying assignment for Π, we will focus on good edges.

If an edge (u, v) is good then

$$\mathop{\mathbb{E}}_{x,y,z} [f_u(x) f_v(y) f_u(x(y \circ \pi_{uv})z)] \geq \left(\frac{1}{2} + \frac{\epsilon}{2}\right) - \left(\frac{1}{2} - \frac{\epsilon}{2}\right) = \epsilon.$$

Following the analysis of the dictatorship test, we expand the left-hand side using the Fourier expansion of all functions involved:

$$\begin{aligned}\epsilon &\leq \sum_{S,T,U} \hat{f}_u(S)\hat{f}_v(T)\hat{f}_v(U) \mathop{\mathbb{E}}_{x,y,z}[\chi_S(x)\chi_T(y)\chi_U(x(y\circ\pi_{uv})z)] \\ &= \sum_{S,T,U} \hat{f}_u(S)\hat{f}_v(T)\hat{f}_v(U) \mathop{\mathbb{E}}_{x}[\chi_S(x)\chi_U(x)] \mathop{\mathbb{E}}_{y}[\chi_T(y)\chi_U(y\circ\pi_{uv})] \mathop{\mathbb{E}}_{z}[\chi_U(z)].\end{aligned}$$

The first expectation equals 1 when $S = U$, and vanishes otherwise. The second expectation equals 1 if T is the set of values appearing an odd number of times in $(\pi_{uv}(\sigma))_{\sigma\in U}$, and vanishes otherwise; we denote this set by $\pi^2_{uv}(U)$. Finally, since $\Pr[z_i = 1] = 1 - \delta$, the third expectation is $(1-2\delta)^{|U|}$. In total, we get

$$\sum_{S\subseteq[\Sigma]} (1-2\delta)^{|S|}\hat{f}_u(S)^2\hat{f}_v(\pi^2_{uv}(S)) \geq \epsilon. \tag{1}$$

At this point, we could apply the analysis of the dictatorship test, concluding that

$$\max_{S\subseteq[\Sigma]} (1-2\delta)^{|S|}\hat{f}_v(\pi^2_{uv}(S)) \geq \epsilon,$$

using Parseval's identity on f_u. We conclude that there is a small set S such that $\hat{f}_v(\pi^2_{uv}(S))$ is large. A reasonable choice of a color for v is a random element of $\pi^2_{uv}(S)$. There are several issues with the idea. First, $\pi^2_{uv}(S)$ could be empty. Second, there could be different choices of $\pi^2_{uv}(S)$ arising from different choices of edges. Third, it is not so clear how to choose a color for u given this information.

Instead, let us make the following observation: since f_u is Boolean, we can think of $\hat{f}_u(S)^2$ as specifying a probability distribution on subsets of $[\Sigma]$, known as the *spectral sample*. The inequality above implies in particular that

$$\sum_{S\subseteq[\Sigma]} (1-2\delta)^{|S|}\hat{f}_u(S)^2 \geq \epsilon,$$

and so the spectral sample is concentrated on small sets. This gives us a way of sampling a color for u which does not involve any specific edge: choose a set S according to the spectral sample of f_u, and then choose a random element $c(u) \in S$, using the fact that S is non-empty since f_u is odd. It is natural to choose a color for v in precisely the same way: choose a set T according to the spectral sample of f_v, and then choose a random element $c(v) \in T$.

What is the probability that this choice satisfies the constraint π_{uv}? If $T = \pi^2_{uv}(S)$ then $c(v) = \pi_{uv}(c(u))$ with probability at least $1/|S|$, since given the choice of $c(v)$, there is at least one element in S which maps to $c(v)$ under π_{uv}. Therefore the probability that $\pi_{uv}(c(u)) = c(v)$ is at least

$$\sum_S \frac{1}{|S|}\hat{f}_u(S)^2\hat{f}_v(\pi^2_{uv}(S))^2. \tag{2}$$

We would like to relate this to inequality (1). The obvious thing to try is to apply the Cauchy–Schwarz inequality, splitting $\hat{f}_u(S)^2$ into two factors:

$$\begin{aligned}\epsilon \leq \sum_S \hat{f}_u(S)\cdot(1-2\delta)^{|S|}\hat{f}_u(S)\hat{f}_v(\pi^2_{uv}(S)) &\leq \sqrt{\sum_S \hat{f}_u(S)^2}\sqrt{\sum_S (1-2\delta)^{2|S|}\hat{f}_u(S)^2\hat{f}_v(\pi^2_{uv}(S))^2} \\ &\leq \sqrt{\sum_S (1-2\delta)^{2|S|}\hat{f}_u(S)^2\hat{f}_v(\pi^2_{uv}(S))^2}.\end{aligned}$$

This is almost in the form we want. The only difference is that (2) contains $1/|S|$ rather than $(1-2\delta)^{2|S|}$. Since $(1-2\delta)^{2|S|}$ decreases much faster than $1/|S|$, we can bound the former by the latter:

$$\frac{(1-2\delta)^{2|S|}}{\frac{1}{|S|}} \leq |S|e^{-4\delta|S|} = \frac{1}{4\delta}\cdot 4\delta|S|e^{-4\delta|S|} \leq \frac{1}{4e\delta},$$

since $xe^{-x} \leq e^{-1}$ for $x \geq 0$. In total, this shows that for every good constraint,

$$\Pr[c(v) = \pi_{uv}(c(u))] \geq 4e\delta \sum_S (1-2\delta)^{2|S|} \hat{f}_u(S)^2 \hat{f}_v(\pi_{uv}^2(S))^2 \geq 4e\delta\epsilon^2.$$

Recalling that the fraction of good constraints is at least $\epsilon/2$, this shows that a random assignment of colors satisfies a $2e\delta\epsilon^3$ fraction of constraints. Hence soundness is satisfied as long as $2e\delta\epsilon^3 > \gamma$.

Recap We want to show that it is NP-hard to distinguish instances of MAX-3LIN with value at least $1-\epsilon$ from those with value at most $1/2+\epsilon$, for arbitrary $\epsilon > 0$. We do this by reduction from Label Cover. Given ϵ, we choose a parameter $\gamma > 0$, depending only on ϵ (we make an explicit choice below). Applying the PCP theorem together with parallel repetition, we get that for some alphabets Σ, Δ depending only on γ, it is NP-hard to distinguish between the following two cases, given an instance Π of (Σ, Δ)-Label Cover:

- **Completeness.** Π is satisfiable.
- **Soundness.** Π has value at most γ.

We reduced Π to an instance Ψ of MAX-3LIN with the following properties:

- **Completeness.** If Π is satisfiable, then Ψ has value $1-\delta$.
- **Soundness.** If Ψ has value more than $1/2+\epsilon$ then Π has value at least $2e\delta\epsilon^3$.

Let us now choose $\delta = \epsilon$ and $\gamma = 5\epsilon^4 < 2e\epsilon^4$. If Π is satisfiable then Ψ has value $1-\epsilon$, and if Π has value at most γ then Ψ has value at most $1/2+\epsilon$. We conclude that it is NP-hard to distinguish instances of MAX-3LIN with value at least $1-\epsilon$ from those with value at most $1/2+\epsilon$.

3.6.1 Variant

Here is a slight variant of the soundness proof. We assume that

$$\Pr_{\substack{\{u,v\}\in E \\ x,y,z}}[f_u(x)f_v(y)f_u(x(y\circ\pi_{uv})z) = 1] > \frac{1}{2} + \epsilon,$$

or equivalently,

$$\mathop{\mathbb{E}}_{\substack{\{u,v\}\in E \\ x,y,z}}[f_u(x)f_v(y)f_u(x(y\circ\pi_{uv})z)] > 2\epsilon.$$

Instead of considering only good edges, we consider all edges. Let

$$\epsilon_{uv} = \mathop{\mathbb{E}}_{x,y,z}[f_u(x)f_v(y)f_u(x(y\circ\pi_{uv})z)], \qquad \mathop{\mathbb{E}}_{\{u,v\}\in E}[\epsilon_{uv}] > 2\epsilon.$$

As in the original analysis,

$$\begin{aligned}
\epsilon_{uv}^2 &= \left(\sum_{S\subseteq[\Sigma]} (1-2\delta)^{|S|} \hat{f}_u(S)^2 \hat{f}_v(\pi_{uv}^2(S))\right)^2 \\
&\leq \sum_{S\subseteq[\Sigma]} (1-2\delta)^{2|S|} \hat{f}_u(S)^2 \hat{f}_v(\pi_{uv}^2(S))^2 \\
&\leq \frac{1}{4e\delta} \Pr[c(v) = \pi_{uv}(c(u))],
\end{aligned}$$

where $c(u)$ is chosen by sampling $S \subseteq [\Sigma]$ with probability $\hat{f}_u(S)^2$ and then a random element of S (note $S \neq \emptyset$ since f_u is odd), and $c(v)$ is chosen in the same way. The expected value of this random coloring is

$$\mathop{\mathbb{E}}_{(u,v)\in E}\big[\Pr[c(v) = \pi_{uv}(c(u))]\big] \geq 4e\delta \mathop{\mathbb{E}}_{(u,v)\in E}[\epsilon_{uv}^2] \geq 4e\delta \mathop{\mathbb{E}}_{\{u,v\}\in E}[\epsilon_{uv}]^2 > 16e\delta\epsilon^2,$$

which is better than the bound $2e\delta\epsilon^3$ obtained above. We can now complete the proof as before.

4 Approximability: MAX-CUT

MAX-CUT is the following fundamental problem: partition the vertices of an undirected graph into two parts, maximizing the number of edges crossing the cut. It will be convenient to allow non-negative weights on the edges. An instance of MAX-CUT is then specified by a triple $G = (V, E, w)$, where $w \colon E \to \mathbb{R}_{\geq 0}$.

We can view MAX-CUT as a CSP with a binary variable x_v for each vertex $v \in V$. The value of x_v encodes which of the two parts of V the vertex v belongs to. An edge $\{u, v\}$ is cut if $x_u \neq x_v$. Therefore MAX-CUT is just an optimization version of CSP($\neq$). Allowing edges to be weighted corresponds to allowing constraints to be weighted. This doesn't affect the complexity of the problem by much, since we can duplicate constraints in proportion to their weight.

4.1 Goemans–Williamson algorithm for MAX-CUT

Karp [Kar72] showed in his classic 1972 paper that the decision version of MAX-CUT is NP-complete. In this section we describe an approximation algorithm due to Goemans and Williamson [GW95], which introduced semidefinite programming into the world of approximation algorithms.

Our starting point is an integer program for MAX-CUT. Recall that we have a Boolean variable x_v for each vertex. It will be convenient to have it range over the two values $1, -1$. An edge $\{u, v\}$ of weight w_{uv} is cut if $x_u \neq x_v$. We can express the corresponding indicator variable as $(1 - x_u x_v)/2$. We reach the following integer program, whose optimal value is exactly the maximum weight of a cut in the graph:

$$\begin{aligned} \max \sum_{\{u,v\} \in E} w_{uv} \frac{1 - x_u x_v}{2} \\ s.t.\ x_v^2 = 1 \text{ for all } v \in V \end{aligned}$$

Integer programs are hard to solve in general: we can easily encode SAT using them. When the objective function is linear in the variables, it is natural to relax the integer program into a linear program. Here, however, the objective function is quadratic in the variables, so instead we relax it into a *semidefinite program*. The idea is to replace each variable by a *vector* of arbitrary dimension:

$$\begin{aligned} \max \sum_{\{u,v\} \in E} w_{uv} \frac{1 - \langle x_u, x_v \rangle}{2} \\ s.t.\ \|x_v\|^2 = 1 \text{ for all } v \in V \end{aligned}$$

In fact, without loss of generality we can assume that the vectors reside in $\mathbb{R}^n$, since given n vectors in Euclidean space we can always project them into an n-dimensional subspace in a way which preserves inner products (in fact, $n - 1$ dimensions suffice).

This "vector program" is a *relaxation* of the original integer program, in the sense that its optimal value can only be larger. The reason is that any solution to the integer program translates to a solution of the vector program: we just map ± 1 to $\pm e$, where e is some arbitrary unit vector.

At this point it is not clear whether we've made any progress: are vector programs any easier to solve than integer programs? It turns out that they can be solved efficiently (up to an arbitrarily small approximation error). The first step is to get rid of the vectors. Consider the symmetric matrix $M_{uv} = \langle x_u, x_v \rangle$ of all inner products. This matrix is *positive semidefinite (PSD)*: for any vector $z \in \mathbb{R}^V$,

$$z^T M z = \sum_{u,v \in V} z_u z_v \langle x_u, x_v \rangle = \sum_{u,v \in V} \langle z_u x_u, z_v x_v \rangle = \left\| \sum_v z_v x_v \right\|^2 \geq 0.$$

Conversely, given an arbitrary (symmetric) PSD matrix M, we can find a Cholesky decomposition $M = XX^T$ which gives us vectors x_v (the rows of X) satisfying $M_{uv} = \langle x_u, x_v \rangle$. Therefore the vector program is

equivalent to the semidefinite program

$$\begin{aligned} \max & \sum_{\{u,v\}\in E} w_{uv}\frac{1-M_{uv}}{2} \\ s.t.\ & M_{vv} = 1 \text{ for all } v \in V \\ & M \succeq 0 \end{aligned}$$

Here $M \succeq 0$ states that M is PSD (which includes being symmetric).

A program whose objective function is linear and whose constraints are either linear inequalities or of the form "$M \succeq 0$" is known as a *semidefinite program (SDP)*. It turns out that semidefinite programs can be solved efficiently, using either the ellipsoid algorithm or various interior-point methods; we won't describe any of these algorithms here.

Rounding Suppose we solved the MAX-CUT semidefinite program, obtaining a vector solution x_v via Cholesky decomposition. How do we translate it into a cut, a process known as *rounding*? If the vectors were one-dimensional (which is the case for linear programming relaxations), then the natural course of action is to put negative values on one side, and positive values on the other side. Here, however, we have vectors, so we first have to project them into a single dimension.

Given one set of vectors satisfying $M_{uv} = \langle x_u, x_v\rangle$, we can find another set of solutions by *rotating* the vectors, that is, replacing x_v by Ux_v, where U is an arbitrary unitary matrix; this is because $\langle Ux_u, Ux_v\rangle = \langle x_u, x_v\rangle$. Therefore the vectors x_v should be thought of as defined only up to an arbitrary rotation. This suggests the following simple algorithm for projecting the vectors into one dimension: rotate the vectors using a random rotation, and project into the first argument.

Denoting the random rotation by U, this projects x_v into $(Ux_v)_1 = \langle z, x_v\rangle$, where z is the first row of U, which is just a random vector on the unit sphere. We call z a *random direction*. We can compute a random direction by sampling independent unit Gaussians $z_1, \ldots, z_n$, and normalizing the resulting vector so that it has unit norm; in fact, in our case we can skip the normalization step, since it doesn't affect the result. The entire rounding process is known as *random hyperplane rounding*, where the hyperplane is $\{x : \langle z, x\rangle = 0\}$. The hyperplane partitions the space into two parts, and we partition V according to which part the vectors x_v belong to.

How good is this solution? It suffices to calculate the probability that an edge $\{u, v\}$ is cut, and compare this probability to its contribution to the objective function, which is $\frac{1-\langle x_u, x_v\rangle}{2}$. To this end, we can project x_u and x_v orthogonally to some two-dimensional plane, and rotate them so that $x_u = (1, 0)$ and $x_v = (\cos\theta, \sin\theta)$; thus $\langle x_u, x_v\rangle = \cos\theta$. The situation then looks as in Figure 1.

If $x_u = x_v$ (and so $\theta = 0$), then $\langle z, x_u\rangle$ and $\langle z, x_v\rangle$ always have the same sign. If $x_u = -x_v$, then $\langle z, x_u\rangle$ and $\langle z, x_v\rangle$ always have opposite signs. (In both cases, we ignore the possibility that the inner product is zero, since this happens with probability zero.) In between, the probability varies linearly with θ, as explained in Figure 1. Therefore the edge $\{u, v\}$ is cut with probability $|\theta|/\pi$.

Sometimes $|\theta|/\pi$ is larger than $(1 - \cos\theta)/2$, and sometimes it is smaller; see Figure 2. We can always say that

$$\frac{\theta}{\pi} \geq \alpha_{GW}\frac{1-\cos\theta}{2}, \text{ where } \alpha_{GW} = \min_{0\leq\theta\leq\pi}\frac{2}{\pi}\frac{\theta}{1-\cos\theta}.$$

We can compute numerically that $\theta_{GW} \approx 2.33$ and $\alpha_{GW} \approx 0.878$. It follows that the expected weight of edges cut by our rounding procedure is

$$\sum_{\{u,v\}\in E} w_{uv}\frac{\cos^{-1} M_{uv}}{\pi} \geq \alpha_{GW}\sum_{\{u,v\}} w_{uv}\frac{1-M_{uv}}{2}.$$

Since our SDP is a relaxation, the expression on the right is at least α_{GW} times the value of the maximum cut in the graph. In other words, this algorithm produces an α_{GW}-approximation.

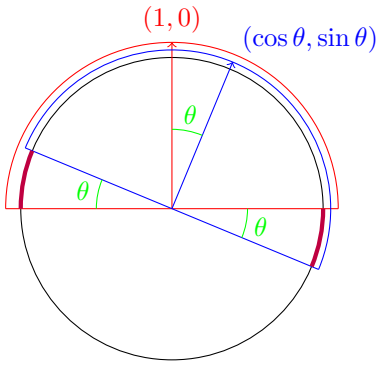


Figure 1: A vector z corresponds to a point on the unit circle. The red region is where $\langle z, x_u \rangle$ is positive, the blue region is where $\langle z, x_v \rangle$ is positive, the purple region is where they have different signs.

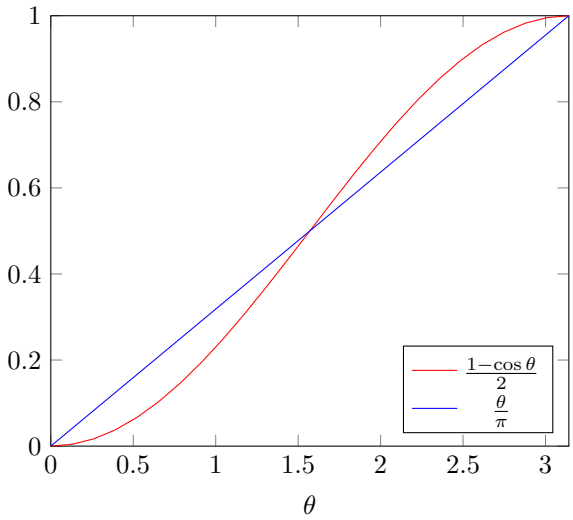


Figure 2: Probability that edge is cut (blue) compared to its contribution to the objective value (red) as a function of the angle.

Before discussing the algorithm any further, let us mention a further issue: the algorithm, as currently described, is randomized. It turns out that it can be derandomized using the method of conditional expectations. Since we cannot go over all possible values of each coordinate of each x_v, this also involves a quantization step. The details were worked out by Mahajan and Ramesh [MR99].

4.2 Algorithmic gap

Our analysis shows that the Goemans–Williamson has an approximation ratio of at least $\alpha_{GW} \approx 0.878$. Is the analysis tight? Can the rounding procedure be improved? Is there a better approximation algorithm possible? We tackle these questions one by one, starting with the tightness of the analysis. We will describe work due to Karloff [Kar99] and Alon–Sudakov [AS00].

Let $\rho_{GW} = \cos\theta_{GW} \approx -0.689$. In order for the analysis to be tight, we need to find an instance whose optimal solution x_v satisfies $\langle x_u, x_v\rangle \approx \rho_{GW}$ for all edges. With hindsight, we will construct such a graph on the vertex set $\{\pm 1\}^n$, for large n; this will enable us to employ techniques from Boolean function analysis. We will attempt to find an instance in which an optimal solution to the SDP is $x_v = v/\|v\| = v/\sqrt{n}$ for all $v \in \{\pm 1\}^n$.

In order to facilitate the analysis, we will construct the instance by specifying a probability distribution on edges which is a *product distribution*, that is, an edge (u, v) is sampled by sampling the coordinates (u_i, v_i) independently, and the weight of an edge is its probability. Experience from Boolean function analysis suggests that product distributions are easier to analyze, due to the product nature of the Fourier basis. Furthermore, it is natural to ask for the marginals to be uniform, that is, u by itself is uniform, and v by itself is uniform.

Recall that we aim for $\langle x_u, x_v\rangle$ to be close to ρ_{GW}. Together with the assumption that the coordinates (u_i, v_i) are identically distributed, this essentially determines the distribution of edges. Since $\langle x_u, x_v\rangle = \langle u, v\rangle/n$, we would like $\mathbb{E}[\langle u, v\rangle] = \rho_{GW} n$. On the other hand, $\mathbb{E}[\langle u, v\rangle] = \mathbb{E}[u_1 v_1]n$, and so we need $\mathbb{E}[u_1 v_1] = \rho_{GW}$. Moreover, in order for the marginals to be uniform, we need $\mathbb{E}[u_1] = \mathbb{E}[v_1] = 0$. There is a unique distribution on pairs of bits which satisfies these constraints: sample $u_1 \in \{\pm 1\}$ uniformly, and let $v_1 = u_1$ with probability $(1 + \rho_{GW})/2$, and $v_1 = -u_1$ with probability $(1 - \rho_{GW})/2$.

If the vectors x_v are an optimal solution to the SDP, then we get what we want: due to the central limit theorem, $\langle x_u, x_v\rangle$ is concentrated on ρ_{GW}, and so the rounding procedure will lose a factor of α_{GW}. It remains to show that the vectors x_v are an optimal solution.

Let f_v be an arbitrary solution to the SDP. We can think of f as a function from $\{\pm 1\}^n$ to $\mathbb{R}^N$, where N is the dimension of the vectors. Accordingly, we will write $f(v)$ for the vector chosen for v. We want to show that the assignment x_v maximizes $\mathbb{E}_{u,v}[(1 - \langle f(u), f(v)\rangle)/2]$, or equivalently, that it minimizes $\mathbb{E}_{u,v}[\langle f(u), f(v)\rangle]$, a quantity which is equal to ρ_{GW} when $f(u) = x_u$.

We will analyze the projection of f to each of its coordinates. The contribution of coordinate i to $\mathbb{E}_{u,v}[\langle f(u), f(v)\rangle]$ is

$$\mathop{\mathbb{E}}_{u,v}[f_i(u) f_i(v)] = \sum_{S,T} \hat{f}_i(S)\hat{f}_i(T) \mathop{\mathbb{E}}_{u,v}[\chi_S(u)\chi_T(v)].$$

We can write $v = uw$, where each coordinate of w equals 1 with probability $(1 + \rho_{GW})/2$ and -1 with

probability $(1-\rho_{GW})/2$. Thus

$$\begin{aligned}
\mathop{\mathbb{E}}_{u,v}[f_i(u)f_i(v)] &= \sum_{S,T} \hat{f}_i(S)\hat{f}_i(T) \mathop{\mathbb{E}}_{u,w}[\chi_S(u)\chi_T(uw)] \\
&= \sum_{S,T} \hat{f}_i(S)\hat{f}_i(T) \mathop{\mathbb{E}}_{u}[\chi_S(u)\chi_T(u)] \mathop{\mathbb{E}}_{w}[\chi_T(w)] \\
&= \sum_{S} \rho_{GW}^{|S|} \hat{f}_i(S)^2 \\
&\geq \rho_{GW} \sum_{S} \hat{f}_i(S)^2 \\
&= \rho_{GW} \mathop{\mathbb{E}}_{v}[f_i(v)^2],
\end{aligned}$$

using $\rho_{GW} \approx -0.689 \in (-1, 0)$. Combining the contribution of all coordinates, we get

$$\mathop{\mathbb{E}}_{u,v}[\langle f(u), f(v)\rangle] = \sum_i \mathop{\mathbb{E}}_{u,v}[f_i(u)f_i(v)] \geq \rho_{GW} \sum_i \mathop{\mathbb{E}}_{v}[f_i(v)^2] = \rho_{GW} \mathop{\mathbb{E}}_{v}[\|f(v)\|^2] = \rho_{GW},$$

since $f(v)$ is a unit vector for all $v \in \{\pm 1\}^N$. Thus $f(v) = x_v$ is indeed an optimal solution to the SDP.

What is the optimal value of this instance? Our calculation above shows that any cut $f\colon \{\pm 1\}^n \to \{\pm 1\}$ satisfies

$$\mathop{\mathbb{E}}_{u,v}[f(u)f(v)] \geq \rho_{GW},$$

with equality if all the Fourier mass of f lies on Fourier coefficients of size 1, that is, f is of the form

$$f(x_1, \ldots, x_n) = \sum_{i=1}^{n} \hat{f}(\{i\}) x_i.$$

The dictator functions $f = \pm x_i$ are of this form; indeed, these are the unique Boolean functions of this form. In other words, if we partition the vertices into two parts according to the first coordinate, then we get a cut whose value is $(1-\rho_{GW})/2$, and this value is optimal; moreover, cuts of this form are the unique optimizers. Thus in this instance, the value of the SDP is the same as the value of the original integer program.

4.3 Integrality gap

So far we have seen that our analysis of random hyperplane rounding is tight. But perhaps there is a better rounding procedure? One way to rule this out is via the following observation: analyses of rounding procedures tend to prove bounds of the form $ALG \geq \alpha SDP$, where ALG is the expected value of the solution produced by the algorithm, and SDP is the value of the SDP. Combined with the trivial observation $SDP \geq OPT$, where OPT is the integer optimum, this shows that the algorithm is an α-approximation.

Since $OPT \geq ALG$, an analysis of this form guarantees that $OPT \geq \alpha SDP$. Therefore, if we construct an instance satisfying $OPT = \beta SDP$, then no analysis of this form can prove an approximation ratio better than β. Such an instance is known as an *integrality gap* instance, since there is a gap between the integer solution OPT and its fractional relaxation SDP. (While we have described integrality gaps for SDP relaxations, they are also used to analyze LP relaxations.)

This suggests looking for an instance of MAX-CUT satisfying $OPT \approx \alpha_{GW} SDP$, following the footsteps of Feige and Schechtman [FS02]. Since $ALG \geq \alpha_{GW} SDP \approx OPT$, in such an instance the Goemans–Williamson algorithm actually constructs a near-optimal solution. Moreover, the rounding procedure needs to lose a factor of α_{GW}, and so again we are looking for a solution satisfying $\langle x_u, x_v\rangle \approx \rho_{GW}$ for all edges (u, v).

In order to compute the value of the SDP, we would like to arrange, just as before, that x_v be an optimal solution for the SDP. However, this time there will be no need to prove that it is optimal. Rather, denoting

its objective value by OBJ, it will suffice to show that $\alpha_{GW} OBJ \approx OPT$, since $\alpha_{GW} SDP \leq ALG \leq OPT \approx \alpha_{GW} OBJ$ will imply that $SDP \lesssim OBJ$ and so $SDP \approx OBJ$.

For the algorithmic gap, we chose as our graph the Boolean cube. This time, we make an even more obvious choice: our graph will consist of a high-dimensional unit sphere! While this is an infinite graph, it can be rounded to a finite one, a step that we skip. We will identify points on the sphere with the corresponding vectors. Once again, it is natural to ask for a probability distribution on pairs of points whose marginals are uniform, and moreover, the inner product is concentrated around ρ_{GW}.

As a stepping stone toward constructing such a distribution, let us start with constructing a uniformly random point on the $(d-1)$-dimensional unit sphere $\mathcal{S}^{d-1}$ (which is the subset of $\mathbb{R}^d$ consisting of vectors of unit norm). The standard technique is to sample a d-dimensional vector x whose coordinates are iid standard Gaussians, and normalize it. This works due to the invariance of the distribution of x under rotation.

Given two such vectors x, y, if we want $\langle x/\|x\|, y/\|y\| \rangle \approx \rho_{GW}$, it seems like a good idea to require $\mathbb{E}[x_i y_i] = \rho_{GW}$; such a pair of Gaussians is said to be ρ_{GW}*-correlated*. There is a unique bivariate Gaussian distribution with this property. One way to sample it is to sample $x_i, z_i \sim N(0,1)$ and to take $y_i = \rho x_i + \sqrt{1-\rho^2} z_i$. Since $\|x\|, \|y\|$ are tightly concentrated around $\sqrt{d}$ (for large d), this will indeed ensure that $\langle x/\|x\|, y/\|y\| \rangle \approx \rho_{GW}$.

Our instance is thus obtained by sampling d-dimensional ρ-correlated Gaussians x, y, and outputting the edge $(x/\|x\|, y/\|y\|)$. The objective value OBJ of the assignment in which each vector is mapped to itself is roughly $(1 - \rho_{GW})/2$. In order to determine OPT, let us observe that each partition of the sphere into two parts translates to a corresponding partition of d-dimensional Gaussian space $\mathbb{R}^d$ into two parts: $x \in \mathbb{R}^d$ is put in the same part as $x/\|x\| \in \mathcal{S}^{d-1}$. We can thus bound OPT by the maximum of

$$\Pr_{x,y}[f(x) \neq f(y)],$$

where x, y are ρ-correlated Gaussians and $f \colon \{\pm 1\}^d \to \{\pm 1\}$.

Recall that our instance will need to be such that the rounding procedure yields an almost optimal solution. What does the rounding procedure do to our solution, which is the uniform distribution on the sphere? It rotates it at random, which has no effect, and then calculates the sign of the first coordinate. When pulling this back to Gaussian space, this corresponds to calculating the sign of x_1, that is, the solution $f(x) = \operatorname{sgn} x_1$.

What is $\Pr[\operatorname{sgn} x_1 \neq \operatorname{sgn} y_1]$? Recall that $y_1 = \rho_{GW} x_1 + \sqrt{1 - \rho_{GW}} z_1$, where $z_1 \sim N(0,1)$. Since $\rho_{GW} = \cos \theta_{GW}$, we can write this as follows:

$$x_1 = \langle (1,0), (x_1, z_1) \rangle, \quad y_1 = \langle (\cos \theta_{GW}, \sin \theta_{GW}), (x_1, z_1) \rangle.$$

Since we are interested only in the signs of x_1 and y_1, we can normalize (x_1, z_1) by its norm to obtain a random direction v. Once we do this, $\operatorname{sgn} x_1$ and $\operatorname{sgn} y_1$ are simply the result of applying random hyperplane rounding to $(1,0)$ and $(\cos \theta_{GW}, \sin \theta_{GW})$! Thus $\Pr[\operatorname{sgn} x_1 \neq \operatorname{sgn} y_1] = \theta_{GW}/\pi$.

Borell's isoperimetric theorem [STs74, Bor75] states (in a special case) that $f(x) = \operatorname{sgn} x_1$ maximizes $\Pr[f(x) \neq f(y)]$, where x, y are ρ_{GW}-correlated Gaussians (Borell's theorem applies for arbitrary ρ; when $\rho > 0$, random hyperplane rounding *minimizes* the stated probability). Borell's theorem can be proved in many ways, some of them quite elementary; but will we just accept it on faith.

Borell's theorem shows that $OPT \leq \theta_{GW}/\pi$; in fact, the Goemans–Williamson algorithm produces a solution with (roughly) this objective value. On the other hand, we have shown above that $SDP \geq (1 - \cos \theta_{GW})/2$. Therefore

$$\frac{OPT}{SDP} \leq \frac{2}{\pi} \frac{\theta_{GW}}{1 - \cos \theta_{GW}} = \alpha_{GW}.$$

This is the desired integrality gap, showing that the Goemans–Williamson rounding procedure is optimal (for arguments which compare ALG only to SDP).

4.4 Hardness of approximation, and the unique games conjecture

We have shown that the analysis of the Goemans–Williamson algorithm is tight, and that random hyperplane rounding cannot be improved. However, this doesn't mean that the Goemans–Williamson algorithm can't be improved, say by using one of the many SDP hierarchies, which are stronger and stronger relaxations, whose value in the limit coincides with the integer program. In this section, we will show (with an asterisk) that this is not the case, following the footsteps of Khot, Kindler, Mossel, and O'Donnell [KKMO07].

We will attempt to reduce Label Cover to MAX-CUT. Given $\epsilon > 0$, we will define $\gamma > 0$, and construct a reduction from the promise version of Label Cover to a gap version of MAX-CUT. Recall that an instance of Label Cover consists of a bipartite graph (U, V, E) and, for each edge, a function $\pi_e \colon \Sigma \to \Delta$, where Σ, Δ are finite alphabets depending only γ. The goal is to construct a Σ-coloring of U and a Δ-coloring of V which satisfies as many of the constraints π_e as possible; denoting the coloring by c, it satisfies a constraint π_{uv} if $\pi_{uv}(c(u)) = c(v)$.

The instance Π of label cover is promised to be either satisfiable or at most γ-satisfiable, that is, at most a γ-fraction of the constraints is satisfied by any coloring. Given such an instance, we wish to construct, in polynomial time, an instance Ψ of MAX-CUT such that for some S, C which can be computed efficiently from Π, the following hold:

- **Completeness.** If Π is satisfiable, then the maximum cut in Ψ is at least C.
- **Soundness.** If Π is at most γ-satisfiable, then the maximum cut in Ψ is at most S.

If S/C is always at most β, then any approximation algorithm whose approximation ratio is better than β can be used to determine whether Π is satisfiable or at most γ-satisfiable, which is NP-hard.

We will describe the instance of MAX-CUT as a Boolean CSP in which the allowed constraints are of the form $x \neq y$. Following our footsteps in the case of MAX-3LIN, we will encode the colors using the Long Code. This means that for every vertex $u \in U$ we will have variables encoding a function $f_u \colon \{\pm 1\}^\Sigma \to \{\pm 1\}$. Similarly, for every vertex $v \in V$ we will have variables encoding a function $f_v \colon \{\pm 1\}^\Delta \to \{\pm 1\}$. In the MAX-3LIN hardness proof, we forced the functions f_u, f_v to be odd using *folding*, which was a requirement of the dictatorship test. Here we cannot do this, since a constraint of the form $x \neq y$ could become a constraint of the form $x = y$.

Dictatorship test Which dictatorship test should we use? One option that comes to mind stems from the construction of the algorithmic gap. Suppose we are given a function $f \colon \{\pm 1\}^n \to \{\pm 1\}$. The construction sampled ρ-correlated $x, y \in \{\pm 1\}^n$ (meaning that the marginals are uniform and $\mathbb{E}[x_i y_i] = \rho_{GW}$), and checked that $f(x) \neq f(y)$. If f is a dictator, say $f(x) = x_i$, then the test passes when $x_i \neq y_i$, which happens with probability $(1 - \rho_{GW})/2 \approx 0.844$.

We are aiming at an inapproximability of α_{GW}, and so the hope is that if $\Pr[f(x) \neq f(y)] \geq \theta_{GW}/\pi + \delta$ then f has some useful structure. When describing the integrality gap, we encountered a problem whose *optimal* solution is θ_{GW}/π, namely maximize $\Pr[f(x) \neq f(y)]$ over ρ-correlated *Gaussians*. We can implement a Gaussian on the Boolean cube using the mapping

$$(x_1, \ldots, x_n) \mapsto \frac{x_1 + \cdots + x_n}{\sqrt{n}}.$$

If the left-hand side is a random point in $\{\pm 1\}^n$, then according to the central limit theorem, the right-hand side is close to a Gaussian (in various metrics). Composing this with the optimal solution for the Gaussian problem, which is the sign function, we obtain the Majority function on the Boolean cube.

How is the Majority function different from a dictator? The Majority function depends "smoothly" on all inputs. One way to quantify this is using *influences*: the influence of the i'th coordinate is the probability that flipping the i'th coordinate flips the value of the function. In the case of dictators, one coordinate has influence 1, and the rest have influence 0. In the case of majority, all coordinates have influence $\Theta(1/\sqrt{n})$, which is the probability that a random point on the cube is exactly balanced.

An important result in Boolean function analysis, the *invariance principle* [MOO10], states that if all influences of a function are small, and an additional mild technical condition holds,[2] then the function behaves as if it lived on Gaussian space! In fact, the invariance principle was proved expressly for completing the analysis of the very result we are now describing: inapproximability of MAX-CUT.

In our case, the invariance principle implies that if $\Pr[f(x) \neq f(y)] \geq \theta_{GW}/\pi + \delta$ then f has an influential coordinate (that is, has influence at least τ, for an appropriate constant τ). While arbitrary Boolean functions can have many influential coordinates (for example, all coordinates of the parity function are maximally influential), we can slightly change the definition of influence to ensure that only $O(1)$ many coordinates can be influential.[3] We are then in much the same position as in the MAX-3LIN reduction.

First attempt and hardness of MAX-2LIN The actual reduction has to combine this dictatorship test with a consistency check that throws in the constraints π_e, but we can do this just as before. Recall that for $y \in \{\pm 1\}^\Delta$, we defined $y \circ \pi_{uv} \in \{-1\}^\Sigma$ as follows: $(y \circ \pi_{uv})_\sigma = y_{\pi_{uv}(\sigma)}$. Our MAX-CUT instance Ψ then corresponds to the following test:

- Choose a random edge $(u, v) \in E$.
- Choose $y \in \{\pm 1\}^\Delta$ at random.
- Choose $z \in \{\pm 1\}^\Delta$ at random in a biased manner: $z_i = 1$ with probability $(1 + \rho_{GW})/2$.
- Check whether $f_v(y) \neq f_u((yz) \circ \pi_{uv})$.

(Previously the noise z was on the Σ side, but here it is more convenient to have it on the Δ side, since there is no x.)

Let's check that everything works out when Π is satisfiable, say using a coloring c. We define $f_u(x) = x_{c(u)}$ and $f_v(y) = y_{c(v)}$. Since $\pi_{uv}(c(u)) = c(v)$, we have

$$f_u((yz) \circ \pi_{uv}) = (yz)_{\pi_{uv}(c(u))} = y_{c(v)} z_{c(v)},$$

and so the test succeeds with probability $(1 - \rho_{GW})/2$, as desired.

Now let us see move on to soundness. Suppose that the test succeeds with probability at least $\theta_{GW}/\pi + \epsilon$. This can only happen if at least $\epsilon/2$ of the edge tests succeed with probability at least $\theta_{GW}/\pi + \epsilon/2$. Call an edge for which this happens a *good* edge. If (u, v) is a good edge then

$$\Pr[f_v(y) \neq F_{u,v}(yz)] \geq \theta_{GW}/\pi + \epsilon/2, \text{ where } F_{u,v}(y) = f_u(y \circ \pi_{uv}).$$

Unfortunately, the probability on the left is not quite what a result such as Borell's theorem actually bounds, since two different functions appear there.

As a warm-up, recall that when analyzing the algorithmic gap, we proved that $\Pr[f(x) \neq f(y)] \leq (1-\rho)/2$ for ρ-correlated inputs x, y on the Boolean cube (where $\rho < 0$); we did this by showing that $\mathbb{E}[f(x)f(y)] \geq \rho$. What happens when we allow two functions f, g? When we repeat the same proof, we arrive at the following identity:

$$\mathbb{E}[f(x)g(y)] = \sum_S \rho^{|S|} \hat{f}(S)\hat{g}(S).$$

If $f = g$ then we can bound $\rho^{|S|}\hat{f}(S)^2 \geq \rho\hat{f}(S)^2$, using the non-negativity of $\hat{f}(S)^2$. However, $\hat{f}(S)\hat{g}(S)$ could be negative. This is reflected in the solution $f = 1$, $g = -1$, which satisfies $\mathbb{E}[f(x)g(y)] = -1$. We can rule out such solutions by declaring f, g to be odd, which can be accomplished by folding. While folding is inadmissible for MAX-CUT, since it could lead to constraints of the form $x = y$, it is admissible for

[2] For the experts: the function has low degree, or at least small Fourier tails. In our case we will ensure this by applying a small amount of noise to the function, which has the effect of slightly changing the value of ρ_{GW}.

[3] For the experts: we can use low-degree influences, or noisy influences.

MAX-2LIN, in which constraints are of either form $x \neq y$ or $x = y$; let's think of MAX-2LIN for now. We can then bound

$$\rho^{|S|}\hat{f}(S)\hat{g}(S) \geq \rho|\hat{f}(S)|\,|\hat{g}(S)|,$$

since there is no contribution from $S \neq \emptyset$ (in fact, it suffices for one of the functions to be balanced). Applying Cauchy–Schwarz,

$$\sum_S |\hat{f}(S)|\,|\hat{g}(S)| \leq \sqrt{\sum_S \hat{f}(S)^2}\sqrt{\sum_S \hat{g}(S)^2},$$

and so we recover the desired inequality $\mathbb{E}[f(x)g(y)] \geq \rho$.

In a similar way, Borell's inequality still works if f, g are odd (it suffices for one of the functions to be balanced). Applying an appropriate version of the invariance principle,[4] and it implies that if Boolean functions f, g satisfy $\mathbb{E}[f(x)g(y)] \geq \theta_{GW}/\pi + \epsilon/2$, then they have a *jointly influential variable*, that is, some variable has large influence *in both* (this will be important when analyzing the soundness of the construction).

Back to our test $f_v(y) \neq F_{u,v}(yz)$. We can guarantee that f_v is balanced by requiring it to be odd, allowing us to apply Borell's theorem. We thus get that f_v and F_u have a jointly influential variable. The next step would be to construct a random coloring. On the V side, this is easy: for each v, we choose a random influential variable. It is less clear what to do on the U side.[5]

Things become much easier if we assume that π_{uv} is a *permutation* (and so $\Sigma = \Delta$). Denoting the set of influential coordinates of f_u by I_u, it is not hard to check that the set of influential coordinates of $F_{u,v}$ is precisely $\pi_{uv}^{-1}(I_u)$, since the i'th input to f_u comes from the $\pi_{uv}(i)$'th input of $F_{u,v}$. Similarly denote the set of influential coordinates of f_v by I_v. If (u, v) is a good edge then according to the combination of Borell's inequality and the invariance principle, the sets $I_u, \pi_{uv}^{-1}(I_v)$ must intersect. Therefore if we color u using a random element $c(u) \in I_u$ and v using a random element $c(v) \in I_v$, we would have $c(v) = \pi_{uv}(c(u))$ with probability $1/M$, where M is a bound on the number of influential coordinates. Consequently, we will satisfy an $\epsilon/2M$-fraction of constraints. This will complete the proof of the soundness of the reduction (for the case of MAX-2LIN!) if $\epsilon/2M > \gamma$.

Can we choose our parameters in such a way that $\epsilon/2M > \gamma$? For that, we need to dig deeper into where M comes from. Recall that M is a bound on the number of influential coordinates, a concept that arises from the invocation of the invariance principle. The invariance principle states that low-influence functions on the Boolean cube behave *almost* as if they lived on Gaussian space. There is a trade-off between M and the quantification of *almost*. The way that *almost* manifests itself is by an additive error, which in our case needs to be smaller than $\epsilon/2$ (recall that Borell's theorem bounds the Gaussian quantity by θ_{GW}/π, and we want to contradict it by assuming that the Boolean quantity is at least $\theta_{GW}/\pi + \epsilon/2$). Crucially, M depends on this additive error *but not on the dimension* $|\Sigma|$; most results in Boolean function analysis have this feature, dimension-independence. Therefore given $\epsilon > 0$, we are free to choose $\gamma < \epsilon/2M$; there is no circular dependence on parameters.

Unique Label Cover and UGC What happens when we restrict the Label Cover constraints π_{uv} to be permutations, a problem known as *Unique Label Cover*? The problem becomes easy, since it is easy to check whether an instance is satisfiable! Indeed, if a coloring satisfies all constraints, then the color of a single vertex determines the colors of all other vertices in its connected component. This leads to a simple algorithm which enumerates the color of one vertex per connected component.

It could still be possible that Unique Label Cover is hard if we relax perfect completeness. That is, the following could still hold: for every $\gamma > 0$ there is an alphabet Σ such that given an instance Π of Unique Label Cover (with $\Delta = \Sigma$), it is NP-hard to distinguish between the following two cases:

[4] For the experts: see [Mos10, Proposition 1.15].

[5] There are two main issues. First, flipping a single coordinate in the input of $F_{u,v}$ corresponds to flipping several coordinates in the input of f_u, and so influential variables of $F_{u,v}$ don't directly correspond to influential variables of f_u. This can be handled by a hybrid argument, incurring a multiplicative loss of $|\Sigma|$. Even worse is that the distribution of the remaining inputs of f_u is not uniform, since some coordinates are identical. This causes a multiplicative loss which is *exponential* in $|\Sigma|$. The argument can perhaps be rescued if we assume that the constraints are 2-to-1, that is, $|\pi_{uv}^{-1}(\tau)| = 2$ for all u, v, τ. This version of Label Cover has recently been proved to be NP-complete, with imperfect completeness [KMS18].

- **Completeness.** The instance Π is at least $(1-\gamma)$-satisfiable.
- **Soundness.** The instance Π is at most γ-satisfiable.

This is the famous *Unique Games Conjecture* (also known as UGC), suggested by Khot [Kho02] as a way to tackle the inapproximability of MAX-2LIN (in a different parameter setting).

We don't know whether the Unique Games Conjecture is true. A major recent result, due to Khot, Minzer and Safra [KMS18] (with help from others), states that the variant of Label Cover in which the constraints satisfy $|\pi_{uv}^{-1}(\tau)| = 2$ for all $\tau \in \Delta$ is NP-hard, with imperfect completeness. Khot conjectured that this version of Label Cover is NP-hard even with perfect completeness, and this conjecture, known as the 2-to-1 conjecture, remains open. Perfect completeness is important in order to get hardness of approximation results with perfect completeness, and to that end, Braverman, Khot and Minzer recently showed that the Unique Games Conjecture is equivalent to a strengthened form of the 2-to-1 conjecture, in which the games in question are "rich" [BKM21]x.

To recap what we have seen so far, we have shown that the unique games conjecture implies that MAX-2LIN is $\alpha_{GW} + \epsilon$-hard to approximate for any $\epsilon > 0$. Unfortunately, no matching approximation algorithm is known.

Hardness of MAX-CUT Let us finally go back to MAX-CUT. Recall that the issue with our reduction is that the analysis involves proving lower bounds on

$$\Pr[f_v(y) \neq F_{u,v}(yz)],$$

where y, z are ρ_{GW}-correlated, and these are only available if we can guarantee that the functions in question are balanced. One way to guarantee this is by forcing the functions to be odd via folding, but this could result in $=$ constraints, which are allowed in MAX-2LIN but not in MAX-CUT. There was no such issue with the single-function version of Borell's theorem.

To gain some intuition, recall that given a coloring c, we defined $f_u(x) = x_{c(u)}$, and so if c satisfies the constraint π_{uv} then $F_{u,v}(y) = y_{\pi_{uv}(c(u))} = y_{c(v)}$. In other words, $F_{u,v}$ depends only on v! This suggests choosing two different vertices $u_1, u_2 \in U$ and comparing $F_{u_1,v}$ and $F_{u_2,v}$. The marginal distributions of $(u_1, v), (u_2, v)$ need to be uniform over all edges in E, and this can be accomplished by first sampling an edge (u_1, v) and then a random neighbor u_2 of v. The probability of choosing an edge (u_2, v) is then

$$\underbrace{\frac{\deg(v)}{|E|}}_{\Pr[v]} \cdot \underbrace{\frac{1}{\deg(v)}}_{\Pr[u_2|v]} = \frac{1}{|E|},$$

as needed. We reach the following test, which specifies an instance Ψ of MAX-CUT:

- Choose a random vertex (u_1, v) and a random neighbor u_2 of v.
- Choose $y \in \{\pm 1\}^{\Delta}$ at random.
- Choose $z \in \{\pm 1\}^{\Delta}$ at random in a biased manner: $z_i = 1$ with probability $(1 + \rho_{GW})/2$.
- Check whether $F_{u_1,v}(y) \neq F_{u_2,v}(yz)$, that is, whether $f_{u_1}(y \circ \pi_{u_1 v}) \neq f_{u_2}((yz) \circ \pi_{u_2 v})$.

Notice that we no longer need to include the functions f_v in the variable set of Ψ.

If the Unique Label Cover instance Π is $(1-\gamma)$-satisfiable, then we can convert a coloring c satisfying a $(1-\gamma)$-fraction of the constraints into a solution to Ψ as follows: $f_u(x) = x_{c(u)}$. The union bound shows that if we choose u_1, u_2, v as in the test, then both constraints $\pi_{u_1 v}, \pi_{u_2 v}$ hold with probability at least $1 - 2\gamma$. Hence Ψ has value at least $C = (1-2\gamma)(1-\rho_{GW})/2$, only slightly smaller than what we obtained for MAX-2LIN.

What about soundness? Consider a solution of Ψ which satisfies an $(\theta_{GW}/\pi + \epsilon)$-fraction of constraints. As before, an $\epsilon/2$-fraction of choices of (u_1, u_2, v) are good in the sense that

$$\Pr_{y,z}[F_{u_1,v}(y) \neq F_{u_2,v}(yz)] \geq \theta_{GW}/\pi + \epsilon/2.$$

However, this expression still involves two different functions. How should we proceed?

Recall that in our intended solution, both functions $F_{u_1,v}, F_{u_2,v}$ were equal to the same dictator $y_{c(v)}$. This suggests a natural way of getting rid of the u-part, namely to consider

$$g_v(y) = \mathop{\mathbb{E}}_{u \in N(v)}[F_{u,v}(y)],$$

where $N(v)$ is the set of neighbors of v. This is no longer guaranteed to be a Boolean function, but it does satisfy $|g_v(y)| \leq 1$, which will suffice for our purposes.

The definition of g_v, which doesn't depend on u_1, u_2, suggests changing the definition of *good*: we will say that $v \in V$ is good if

$$\Pr_{\substack{u_1,u_2 \\ y,z}}[F_{u_1,v}(y) \neq F_{u_2,v}(yz)] \geq \theta_{GW}/\pi + \epsilon/2.$$

The same averaging argument shows that this still happens with probability at least $\epsilon/2$, that is, the fraction of good v is at least $\epsilon/2$.

If v is good, what does this say about g_v? Using the formula $\Pr[F_{u_1,v} \neq F_{u_2,v}] = \mathbb{E}[(1 - F_{u_1,v}F_{u_2,v})/2]$, we see that

$$1 - 2\theta_{GW}/\pi - \epsilon \leq \mathop{\mathbb{E}}_{\substack{u_1,u_2 \\ y,z}}[F_{u_1,v}(y)F_{u_2,v}(yz)] = \mathop{\mathbb{E}}_{yz}[g_v(y)g_v(z)].$$

Borell's theorem was stated for Boolean functions, but in fact it works under the assumption that the function in question takes values in the *interval* $[-1, 1]$; this is easy to check for the corresponding bound on the Boolean cube, which in fact works as long as the norm of the function is at most 1. It therefore follows (via the invariance principle, in its classical form) that g_v has an influential variable, for the correct definition of influence.

Recall that for *Boolean* functions f, we defined the influence of the i'th coordinate as the probability that $f(x) \neq f(x^{\oplus i})$, where $x^{\oplus i}$ results by flipping the i'th coordinate of x. The analogous definition for arbitrary functions is

$$\mathrm{Inf}_i[f] = \mathop{\mathbb{E}}_{x}\left[\left(\frac{f(x) - f(x^{\oplus i})}{2}\right)^2\right].$$

This agrees with the definition for Boolean functions. In fact, any definition of the form $\mathbb{E}[|(f(x)-f(x^{\oplus i}))/2|^p]$ would. The advantage of this particular definition is that it results in a nice formula for the influence. The idea is to expand the expression inside the brackets using the Fourier expansion:

$$\frac{f(x) - f(x^{\oplus i})}{2} = \sum_S \hat{f}(S)\frac{\chi_S(x) - \chi_S(x^{\oplus i})}{2}.$$

If $i \notin S$ then $\chi_S(x)$ doesn't depend on x_i, and so the fraction on the right vanishes. Otherwise, $\chi_S(x^{\oplus i}) = -\chi_S(x)$, and so the fraction on the right evaluates to $\chi_S(x)$. In other words,

$$\frac{f(x) - f(x^{\oplus i})}{2} = \sum_{i \in S} \hat{f}(S)\chi_S(x).$$

The right-hand side is just the Fourier expansion of the left-hand side, and so Parseval's identity shows that

$$\mathrm{Inf}_i[f] = \sum_{i \in S} \hat{f}(S)^2.$$

Remark We can now explain how to bound the number of influential variables. Let $\mathrm{Inf}[f] = \sum_i \mathrm{Inf}_i[f]$, the *total influence* of f, which is equal to $\sum_S |S|\hat{f}(S)^2$. One way is to ask that f have bounded degree d, meaning that $\hat{f}(S) \neq 0$ only when $|S| \leq d$. In this case $\mathrm{Inf}[f] \leq d$, and so there are at most d/τ coordinates whose influence is at least τ. The invariance principle still holds if we only consider the low-degree influences of f, which are the influences of its low-degree part $f^{\leq d} = \sum_{|S| \leq d} \hat{f}(S)\chi_S$.

Another way is to consider the noisy version of f, defined as follows: $T_\rho f(x) = \mathbb{E}[f(y)]$, where (x, y) are ρ-correlated. The total influence of $T_\rho f$ is at most $\max_{s \in \mathbb{N}} s\rho^s$. The invariance principle still holds if we only consider the noisy influences of f, which are the influences of $T_\rho f$ for ρ close to 1.

We now proceed as in the case of MAX-2LIN. Let J_v be the set of influential variables of g_v, which has size at most M. We choose a color $c(v) \in J_v$ at random. It is natural to define J_u in the same way (perhaps with a different threshold), and to color the vertices in U by choosing $c(u) \in J_u$ at random. To see whether this works, let us try to relate $\mathrm{Inf}_i[g_v]$ to the influences of f_u.

The starting point is the Fourier expansion of g_v:

$$g_v(y) = \mathop{\mathbb{E}}_{u \in N(v)}[f_u(y \circ \pi_{u,v})] = \mathop{\mathbb{E}}_{u \in N(v)}\left[\sum_S \hat{f}_u(S)\chi_{\pi_{u,v}(S)}(y)\right] = \sum_T \chi_T(y) \mathop{\mathbb{E}}_{u \in N(v)}[\hat{f}_u(\pi_{u,v}^{-1}(T))].$$

Now suppose that $\mathrm{Inf}_i[g_v] \geq \tau$, where τ is our threshold for *influential* for the sake of defining J_v, which depends only on ϵ. Substituting the formula for the influence,

$$\tau \leq \sum_{i \in T} \left(\mathop{\mathbb{E}}_{u \in N(v)}[\hat{f}_u(\pi_{u,v}^{-1}(T))]\right)^2.$$

Looking forward, we would like to move the square inside the expectation, to get an expression involving squares of Fourier coefficients. We can accomplish this by applying Cauchy–Schwarz:

$$\tau \leq \sum_{i \in T} \mathop{\mathbb{E}}_{u \in N(v)}[\hat{f}_u(\pi_{u,v}^{-1}(T))^2] = \mathop{\mathbb{E}}_{u \in N(v)}\left[\sum_{i \in T} \hat{f}_u(\pi_{u,v}^{-1}(T))^2\right].$$

The expression inside the expectation on the right is at most 1 by Parseval's identity. Applying an averaging argument, wee see that at least a $\tau/2$-fraction of the neighbors $u \in N(v)$ satisfy

$$\sum_{i \in T} \hat{f}_u(\pi_{u,v}^{-1}(T))^2 \geq \frac{\tau}{2}.$$

We call such a neighbor *i-good*. Observe now that

$$\sum_{i \in T} \hat{f}_u(\pi_{u,v}^{-1}(T))^2 = \sum_{i \in \pi_{uv}(S)} \hat{f}_u(S)^2 = \mathrm{Inf}_{\pi_{uv}^{-1}(i)}[f_u].$$

This gives us the correct definition of J_u: it consists of all variables whose influence on f_u is at least $\tau/2$, of which there are at most $2M$ (in fact, in order to be able to obtain this bound, we need to replace influence by one of its variants; this works out). The argument above shows that if $i \in J_v$ then $\pi_{uv}^{-1}(i) \in J_u$. We define $c(u)$ to be a random element of J_u if J_u is non-empty, and arbitrarily otherwise.

Suppose that v is a good vertex and that u is a $c(v)$-good neighbor of v. Then $c(u) = \pi_{uv}^{-1}(c(v)) \in J_u$, and so the constraint π_{uv} is satisfied with probability at least $1/|J_u| = 1/2M$. Since a random vertex is good with probability at least $\epsilon/2$, and a random neighbor is good with probability at least $\tau/2$, a random assignment satisfies an $\epsilon\tau/8M$-fraction of constraints. Since $\epsilon\tau/8M$ depends only on ϵ, we can choose $\gamma < \epsilon\tau/8M$, completing the soundness proof.

Recapping, we gave a reduction that transforms an instance Π of Unique Label Cover into an instance Ψ of MAX-CUT with the following properties:

- **Completeness.** If Π is at least $(1-\gamma)$-satisfiable then Ψ has value at least $(1-2\gamma)(1-\rho_{GW})/2$.

- **Soundness.** If Ψ has value at least $(\theta_{GW}/\pi + \epsilon)$ then Π is more than $\epsilon\tau/8M$-satisfiable, where τ, M depend only on ϵ.

If we choose $\gamma = \epsilon\tau/8M$, then the unique games conjecture implies that it is NP-hard to distinguish between instances of MAX-CUT whose value is at least $C = (1 - \epsilon\tau/4M)(1 - \rho_{GW})/2$ and instances whose value is at most $S = \theta_{GW}/\pi + \epsilon$; we also say that the task is *UGC-hard*. As $\epsilon \to 0$, the ratio between C/S approaches α_{GW}. We conclude that it is UGC-hard to approximate MAX-CUT better than $\alpha_{GW} + \delta$, for any $\delta > 0$.

4.5 Extensions: O'Donnell–Wu and Raghavendra's theorem

The performance of the Goemans–Williamson algorithm depends on the value of the true maximum cut: for example, one can show using our analysis that if the maximum cut has value $1 - \epsilon$, then the algorithm finds a cut whose value is $1 - O(\sqrt{\epsilon})$. Khot introduced his unique games conjecture to show that this is tight, up to the hidden constant. More generally, for a given optimization problem, we can study the set of pairs (C, S) such that it is possible to efficiently distinguish between instances whose value is at least C and those whose value is at most S. For MAX-CUT, this was accomplished by O'Donnell and Wu [OW08] (assuming the unique games conjecture).

Algorithmically, O'Donnell and Wu use the same SDP, rounded by projecting to a random direction and then probabilistically rounding the result to ± 1 in an arbitrary way (rather than just using the sign, as in the Goemans–Williamson algorithm). To prove hardness, they first generalize the integrality gap construction of Feige and Schechtman by choosing the correlation ρ at random. They show that the two problems, finding the best rounding procedure and finding the best distribution of ρ, are dual, and so their algorithm matches their integrality gap. Finally, they plug the integrality gap construction into a hardness proof similar to ours, obtaining a matching hardness of approximation result.

Raghavendra [Rag08] extended the analysis in a different direction: to arbitrary CSPs on a fixed finite alphabet. First, Raghavendra constructs a canonical semidefinite program which works for all CSPs. Second, he gives an optimal rounding procedure, involving projection into a constant number of random directions, which matches the optimal integrality gap. Finally, he plugs the integrality gap construction into a hardness of approximation proof, showing that his approximation algorithm is optimal (assuming the unique games conjecture). This reduction differs from ours since it involves more than two functions.

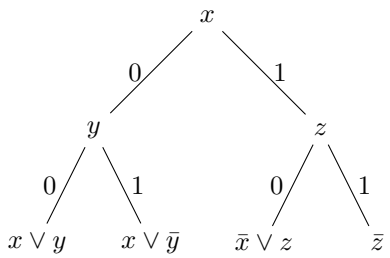
Raghavendra's theorem highlights the importance of both the unique games conjecture and semidefinite programming, and is the crowning jewel in the hardness of approximation literature.

5 Proof complexity

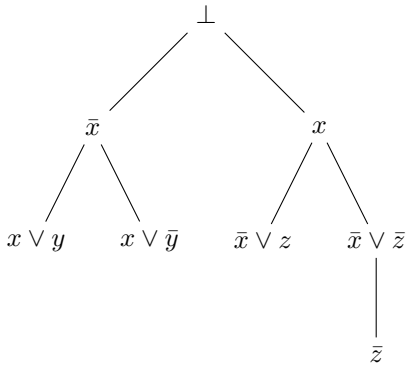
5.1 SAT solving and Resolution

Given a SAT instance, that is, a set of clauses, how do we check whether it is satisfiable or not? One possible algorithm is to try all possible truth assignments, but this always runs in time 2^n, where n is the number of variables. An improvement over this trivial algorithm is an algorithm based on *branching*. At each step, we choose a variable x, and split on its possible truth values. Each branch in this tree corresponds to a partial assignment of the variables. If a partial assignment falsifies a clause then it can be *pruned*. Eventually, either a satisfying assignment is found, or all branches are pruned, in which case we have a decision tree that, given a truth assignment, finds a clause it falsifies.

Here is an example of this algorithm, when run on the CNF $(x \vee y) \wedge (x \vee \bar{y}) \wedge (\bar{x} \vee z) \wedge \bar{z}$:



If all branches below a node v are pruned, then this means that the partial assignment corresponding to v has been ruled out. For example, the node labelled y above rules out the assignment $x = 0$, and this can be expressed as the clause $\bar{x}$. In the case of the rightmost leaf, the ruled out assignment is $x = z = 1$, and the corresponding clause $\bar{x} \vee \bar{z}$ follows from the stated clause. Finally, the root rules out the empty clause, which we denote by $\perp$. We obtain the following tree:

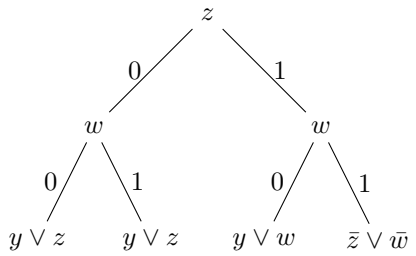


Here each of the leaves is labelled by an input clause, which we can think of as an *axiom*. The clause $\bar{x} \vee \bar{z}$ is obtained from the axiom $\bar{z}$ through *weakening*: given a clause C, we can always derive a clause $C \vee \ell$, for any literal ℓ. Branching corresponds to the *cut rule*: From $C \vee x$ and $D \vee \bar{x}$, we can derive the clause $C \vee D$. We also say that $C \vee D$ is obtained by *resolving* the two clauses $C \vee x$ and $D \vee \bar{x}$. We have reached the proof system known as *Resolution*.

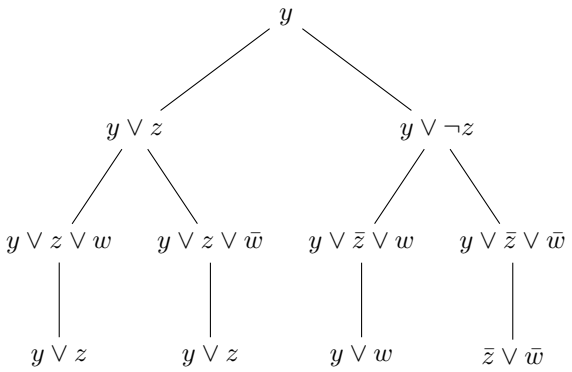
A proof in Resolution is a sequence of *lines*, each of which is a clause. Each line is either an axiom, or follows from a preceding line via weakening, or follows from two preceding lines via the cut rule. Each line logically follows from the lines implying it, and so every line in Resolution logically follows from the axioms.

We often use Resolution as a system for refuting CNFs. The set of clauses comprising the CNF are the axioms. If we can derive the empty clause $\perp$, then we have refuted the CNF. This begs the following question: can Resolution be used to refute every unsatisfiable CNF? We can use the connection between Resolution and branching algorithms to prove this in a strong form known as *implicational completeness*.

Suppose that ϕ is a CNF over the variables $x_1, \ldots, x_n$ which implies a clause C, say $C = x_1 \vee \cdots \vee x_\ell$. We can construct a decision tree of depth $n - \ell$ which simply queries all variables $x_{\ell+1}, \ldots, x_n$. Each leaf assigns a truth value to all variables $x_{\ell+1}, \ldots, x_n$. If we complete it to a truth assignment of all variables by setting $x_1, \ldots, x_\ell$ to false, then we must falsify one of the clauses, by our assumption that the CNF implies C. Here is how this works with the CNF $(y \vee z) \wedge (y \vee w) \wedge (\bar{z} \vee \bar{w})$ and the clause y, corresponding to ruling out the partial truth assignment $y = 0$:



We can convert this into a Resolution proof just as before, adding y to each clause:



This process can always be performed, and it shows that if a CNF ϕ implies a clause C, then C can be deduced from ϕ using Resolution.

Admissibility of weakening Do we really need the rule of weakening? In some sense, we do. Consider the CNF $x \wedge y$. This CNF implies the clause $x \vee y$, but the only way to derive it is via weakening. However, the weakening rule can be avoided if all we are interested in is proving unsatisfiability of CNFs. This follows from the following simple observation, in which C' is a *strengthening* of C if it is obtained from C by dropping some of the literals:

Lemma. Suppose that E is the resolvent of C, D, and C', D' are strengthenings of C, D.
Either the resolvent of C', D' is a strengthening of E, or one of C', D' is itself a strengthening of E.

Given a proof consisting of clauses $C_1, \ldots, C_N$ and a strengthening C'_1 of C_1, we can apply the lemma step-by-step to construct a new proof $C'_1, \ldots, C'_N$ in which each C'_i is a strengthening of C_i; some of the steps now consist just of copying clauses which have already been proven. If C_N is the empty clause then so is C'_N, and so applying this step to a refutation of a CNF results in another refutation.

Starting with a refutation of a CNF ϕ, we can apply this transformation repeatedly to get rid of all weakening steps, thus obtaining a refutation of ϕ without weakening steps. Similarly, if we have a proof of C, then we obtain a proof of some strengthening of C without weakening steps. This is the form of implicational completeness which holds without weakening.

The proof of the lemma itself is quite simple. Suppose that E is obtained by cutting over x, say x appears positively in C and negatively in D. If C' contains x and D' contains $\bar{x}$, then cutting C', D' over x results in

a strengthening of E. If C' doesn't contain x then it is a strengthening of E. Similarly, if D' doesn't contain $\bar{x}$ then it is a strengthening of E.

5.2 DPLL and Horn SAT

The DPLL algorithm [DLL62], named after Davis, Putnam, Logemann and Loveland, and itself an improvement of the Davis–Putnam procedure [DP60], improves over the naive branching algorithm by introducing two optimizations: pure literal elimination, and unit propagation. DPLL is the basis for all modern SAT solvers, together with the more recent technique of *clause learning*, which is described below.

Pure literal elimination is the following simple rule: if all occurrences of a variable x are of the same polarity, say all of them are positive, then we might as well assign x to be true, and remove all clauses containing x. If the CNF is still unsatisfiable, then we can still deduce this using Resolution, by simply never using any clause mentioning x.

Unit propagation is an optimization aimed at *unit clauses*, which are clauses containing a single literal. Suppose that our CNF contains a clause x. This means that x must be set to true. Any other clause that contains x is automatically satisfied, and if a clause contains $\bar{x}$, then we can just remove it. The latter operation could create a new unit clause, and we continue the process of propagation until it stabilizes. Since we can remove $\bar{x}$ from a clause by cutting it with the unit clause x, any resulting refutation can still be described using Resolution.

We have actually already seen unit propagation when we discussed Horn SAT in Section 2.7. The algorithm we gave there consisted of a single run of unit propagation (with a slight difference: we only considered positive unit clauses). If this produces an empty clause then the instance is unsatisfiable. Otherwise, the instance contains no unit clauses. Since each remaining clause is still a Horn clause (contains at most one positive literal), assigning false to all remaining variables satisfies the instance.

In the DPLL algorithm, both of these rules are applied at each step, before splitting: first we perform unit propagation, and then we perform pure literal elimination. At this point we can either point at a falsified axiom, or we need to choose a new variable to split over. This choice isn't specified by the algorithm, and different SAT solvers implement it in different ways.

This issue highlights a gap between proof systems and SAT solving algorithms: SAT solvers need to find a short proof *efficiently*. In proof complexity, the goal is to prove lower bounds on the lengths of proofs in a given proof system, such as Resolution. If a certain CNF is hard to prove in Resolution, then no SAT solver which can be simulated by Resolution will refute it efficiently. The other direction isn't so clear. First, SAT solvers cannot necessarily simulate arbitrary Resolution proofs; we will soon see an example of that. Second, even if a given proof could in principle be found by a SAT solver, it is not clear how to implement such a proof search algorithm efficiently, even if we allow ourselves to produce suboptimal but still non-trivial proofs. We address this issue below as well.

CDCL When simulating the variable splitting algorithm using Resolution, the resulting proof corresponds to a decision tree. This means that every derived clause is used at most once. We call such a proof *tree-like*. However, our description of the Resolution proof system contained no such constraint; its proofs could reuse intermediate clauses, resulting in *DAG-like* proofs.

DAG-like proofs are provably stronger than tree-like proofs, though we would not prove so here; see Iwama and Miyazaki [IM99] for an example using the pigeonhole principle, and Ben-Sasson, Impagliazzo and Wigderson [BSIW04] for an example using pebbling formulas. This shows that DPLL cannot possibly simulate Resolution, since it only ever produces tree-like proofs.

In the 1990's, Marques-Silva and Sakallah [MSS99] and, separately, Bayardo and Schrag [BS97] suggested an enhancement to DPLL which causes it to correspond to DAG-like proofs, known as *conflict-driven clause learning* (CDCL). The basic idea is that every time that the algorithm prunes a branch, it *learns* a clause, by analyzing the "conflict" which resulted in the pruning. This is a clause which is implied by the CNF, and suffices to rule out the partial assignment under consideration. The clause is added to the collection of axioms, and can be used to prune other branches. Since a learned clause could be used more than once,

when converting the transcript of a CDCL algorithm to Resolution, the result is a DAG-like proof. CDCL algorithms incorporate further optimizations to the branching process such as *backjumping* (backtracking more than the minimal number of levels), which we will not describe here.

A long line of work, culminating in Pipatsrisawat–Darwiche [PD11], showed that CDCL, as a meta-algorithm, simulates Resolution (with a polynomial loss); see also [BKS04, AFT11]. However, CDCL can only accomplish this nondeterministically: more recently, Atserias and Müller [AM20] showed that it is NP-hard to distinguish between formulas which can be refuted in polynomial length and those which require subexponential length. In particular, Resolution is not *automatable*: no algorithm can produce a Resolution proof of a CNF in time which is polynomial in the shortest length of a Resolution proof of the CNF.

5.3 2SAT and width

Schaefer's theorem, in its classical statement (see Section 2.7), describes three non-trivial tractable families of CSPs: 2SAT, Horn SAT, and linear equations. In the preceding section, we explained how unit propagation can refute any unsatisfiable instance of Horn SAT. Let us now turn to 2SAT.

The main observation is that the resolvent of two width 2 clauses is another width 2 clause: if we resolve $\ell_1 \vee x$ with $\ell_2 \vee \bar{x}$, we obtain $\ell_1 \vee \ell_2$. Since Resolution is complete, this suggests a simple algorithm for solving 2SAT: apply the cut rule to pairs of clauses until no new clauses can be produced in this way. If the empty clause is reached, then the instance is unsatisfiable. Conversely, since Resolution is implicationally complete, if the instance is unsatisfiable, then the empty clause will be reached. Since there are only $O(n^2)$ possible clauses, this algorithm is efficient, running in time $O(n^4)$ or better.

This suggests studying the *refutation width* of a CNF. This is the minimum width of a refutation of the CNF, where the width of a refutation is simply the maximum width of a clause in the refutation. If a CNF has refutation width w, then there are $O(n^w)$ possible clauses, and so the CNF can be refutated in time $O(n^{2w})$, which is efficient (polynomial) if w is constant. Compare this to the result of Atserias and Müller mentioned above, which states that without any assumption on the CNF, Resolution isn't automatable.

5.4 Tseitin contradictions

We will begin our study of width with the so-called *Tseitin contradictions* [Tsei68], one of the most important contradictions in proof complexity (together with the pigeonhole principle). Given a graph $G = (V, E)$ and an assignment $w \colon V \to \{0, 1\}$, the Tseitin formula is a formula in which there is a variable x_e for each edge, and 2^{d-1} clauses for every vertex $v \in V$ of degree d, expressing the linear constraint

$$\bigoplus_{\{u,v\} \in E} x_{uv} = w(v).$$

We always assume that G is connected, since if a Tseitin formula is unsatisfiable, then the subformula corresponding to one of the connected components will also be unsatisfiable, and a refutation could concentrate on it. Furthermore, typically G has constant degree, guaranteeing that the Tseitin contradiction has a linear number of clauses (in terms of the number of variables).

If $\bigoplus_{v \in V} w(v) = 1$ then the formula is a contradiction, since $\bigoplus_{v \in V} w(v)$ is also what we get if we XOR all linear constraints. The converse also holds: if $\bigoplus_{v \in V} w(v) = 0$ then the formula is satisfiable. To see this, let T be a spanning tree of G, and root it at some arbitrary vertex. Given an assignment $w \colon V \to \{0, 1\}$ and an edge uv, where u is closer to the root, let x_{uv} be the XOR of $w(p)$ over all vertices p in the subtree T_v rooted at v (including v).

If v is any vertex other than the root, let u be its parent, and let $v_1, \ldots, v_\ell$ be its children (if any). Then

$$x_{uv} \oplus \bigoplus_{i=1}^{\ell} x_{vv_i} = \bigoplus_{p \in T_v} w(p) \oplus \bigoplus_{i=1}^{\ell} \bigoplus_{p \in T_{v_i}} w(p) = w(v),$$

since every vertex other than $w(v)$ appears twice in the second expression above. If v is the root then

$$\bigoplus_{i=1}^{\ell} x_{vv_i} = \bigoplus_{i=1}^{\ell} \bigoplus_{p \in T_{v_i}} w(p) = \bigoplus_{p \neq v} w(p) = w(v),$$

since $\bigoplus_{v \in V} w(v) = 0$. Therefore x is a satisfying assignment.

Width in "XOR Resolution" Gaussian elimination can be used to refute a Tseitin contradiction, that is, an unsatisfiable Tseitin formula. This can be expressed as a version of Resolution in which the lines are linear equations, and in which the only deduction rule takes two lines $\ell_1 = b_1$ and $\ell_2 = b_2$ and deduces $\ell_1 \oplus \ell_2 = b_1 \oplus b_2$. The goal is to reach $0 = 1$.

It is natural to define the width of a line to be the number of variables appearing in it. The width of a proof is then the maximum width of a line appearing in it, and the width of a contradictory set of linear equalities is the minimum width required to refute it.

Each line in an XOR Resolution proof is an XOR of axioms. In the case of Tseitin formulas, the XOR of the linear equations corresponding to the vertices in $S \subseteq V$ states that the parity of all edges crossing the cut $(S, \overline{S})$ is $\bigoplus_{v \in S} w(v)$; in particular, the width of the line is the size of the cut $(S, \overline{S})$.

If we choose G to be a random 3-regular graph, then with high probability, the size of a cut $(S, \overline{S})$ will be proportional to $\min(|S|, |\overline{S}|)$; this is since such graphs are *expanders*. Explicit constructions of expanders are known. For example, if p is a prime, we can construct a graph whose vertex set is $\mathbb{Z}_p$, connecting $i \in \mathbb{Z}_p$ to $i \pm 1, i^{-1}$, where $0^{-1} = 0$ is a self-loop.

Let G be a (connected) graph on n vertices such that the size of a cut $(S, \overline{S})$ is $\Omega(\min(|S|, |\overline{S}|))$. If we could show that every refutation of a Tseitin contradiction on G must involve a "complex" line, which results from XORing many axioms, then it would follows that the Tseitin contradiction has large width. This suggests tracking, for each line, the number of axioms which XOR to it; we denote this measure by $m(\ell)$.

If ℓ is an axiom then $m(\ell) = 1$. In contrast, $m(0 = 1) = n$. Indeed, if $S \subsetneq V$ then the corresponding axioms are not contradictory. To see this, take any $v \notin S$, and notice that if we flip $w(v)$ then the Tseitin formula becomes satisfiable; in particular, the axioms corresponding to S are not contradictory.

If ℓ is derived from ℓ_1 and ℓ_2 then clearly $m(\ell) \leq m(\ell_1) + m(\ell_2)$, and so the value of m cannot "jump". In particular, we can find a line ℓ in the proof whose m-complexity is between $n/4$ and $n/2$ using a proof search, as we show below. The width of this line is $\Omega(n/4) = \Omega(n)$.

The proof search starts at the contradiction $0 = 1$. As long as the current line ℓ' has m-complexity at least $n/2$, we consider the two lines ℓ'_1, ℓ'_2 implying it, and proceed to the one with larger m-complexity. The search has to stop eventually, since all lines have m-complexity 1 (here we need to assume that $n \geq 3$). When the search stops at a line ℓ, the preceding line ℓ' has m-complexity at least $n/2$, and so $m(\ell) \geq m(\ell')/2 \geq n/4$.

Width in Resolution The same argument can be adapted to bound the width of Resolution refutations of Tseitin contradictions of graphs with large cuts. For a clause C, we define $m(C)$ to be the minimum number of vertices whose corresponding axioms imply C. As before, the m-complexity of axioms is 1 and the m-complexity of the empty clause is n, and so there is a clause C whose m-complexity is between $n/4$ and $n/2$. It remains to show that C has large width.

Suppose that S is a minimal set of vertices whose axioms imply C. It is natural to expect C to mention all variables in the cut $(S, \overline{S})$, and so have width $\Omega(n)$. To see that this is the case, suppose that $\{u, v\}$ is such an edge, say $v \in S$ and $u \notin S$. Suppose that x_{uv} does not appear in C. Since $S \setminus \{v\}$ does not imply C, there is an assignment which satisfies all axioms concerning vertices in $S \setminus \{v\}$ but falsifies C. Such an assignment necessarily satisfies $\bigoplus_{z \sim v} x_{zv} \neq w(v)$. Since x_{uv} does not appear in C, if we flip the value of x_{uv} then we get an assignment which satisfies all axioms concerning vertices in S but still falsifies C, contradicting the assumption.

Concluding, a Tseitin contradiction corresponding to an expander on n vertices requires width $\Omega(n)$ to refute in Resolution. But does this imply that the contradiction is hard to refute under a more standard measure, such as the number of lines?

5.5 Size and width

Ben-Sasson and Wigderson [BSW01], adapting prior work of Clegg, Edmonds and Impagliazzo [CEI96], showed that CNFs which have large Resolution width require long Resolution refutations. In particular, Tseitin contradictions corresponding to bounded degree expanders on n vertices require $2^{\Omega(n)}$ lines to refute in Resolution. This shows that in contrast to the other tractable cases in Schaefer's theorem, refuting contradictory sets of linear equations is hard for Resolution.

Branching and width In order to refute a CNF ϕ, the branching algorithm described in Section 5.1 chooses a variable x and refutes $\phi|_{x=0}$ and $\phi|_{x=1}$. In the context of the branching algorithm, a refutation of $\phi|_{x=0}$ corresponds to a proof of x, and a refutation of $\phi|_{x=1}$ corresponds to a proof of $\bar{x}$. Applying the cut rule, we obtain the empty clause, thus showing that ϕ is unsatisfiable.

This process can be carried out in general. First, let us explain what we mean by $\phi|_{x=0}$. It is the CNF obtained by removing all clauses containing $\bar{x}$ (these clauses are trivially true), and removing x from all clauses containing it (since $x = 0$). Given a refutation of $\phi|_{x=0}$, we can reintroduce x to the clauses from which it was dropped. The result is either a refutation of $\phi|_{x=0}$ or a proof of x. In terms of width, this shows that

$$w(\phi \vdash x) \leq w(\phi|_{x=0} \vdash \bot) + 1,$$

where $w(\phi \vdash C)$ is the minimum width required to deduce C from the clauses in ϕ.

We can improve on this simulation if the clause $\bar{x}$ is already available. Instead of reintroducing x, we simply derive the clauses of $\phi|_{x=0}$ from the clauses of ϕ: to derive C from $C \vee x$, we simply cut it with $\bar{x}$. The final result is the empty clause. This shows that

$$w((\phi \wedge \bar{x}) \vdash \bot) \leq \max(w(\phi|_{x=0} \vdash \bot), w(\phi)),$$

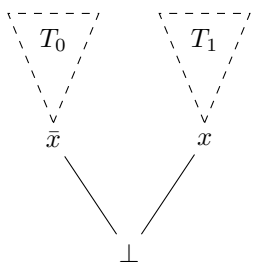
where $w(\phi)$ is the width of ϕ; it comes up when deriving the clauses of $\phi|_{x=0}$.

Putting both kinds of proofs together, we deduce

$$w(\phi \vdash \bot) \leq \max(w(\phi_{x=0} \vdash \bot) + 1, w(\phi_{x=1} \vdash \bot), w(\phi)).$$

This results from first converting a refutation of $\phi|_{x=0}$ to a proof of x, and then using it to convert a refutation of $\phi|_{x=1}$ into a refutation of ϕ. The corresponding refutation is generally not tree-like even if we start with tree-like refutations of $\phi|_{x=0}$ and $\phi|_{x=1}$, since the derived clause $\bar{x}$ is cut against each of the axioms of ϕ containing x.

Tree-like Resolution The branching algorithm always produces a tree-like proof. Can we view every tree-like proof as an application of the branching algorithm? Here is how a general tree-like refutation of a CNF ϕ looks like:



We can convert T_1 into a refutation of $\phi|_{x=0}$ by substituting $x = 0$ throughout the proof. This means that we drop all lines containing the literal $\bar{x}$, and remove x from all lines containing it. This results in a Resolution refutation of $\phi|_{x=0}$. Indeed, axioms of ϕ are transformed into axioms of $\phi|_{x=0}$, and the final line in the new proof is $x|_{x=0} = \bot$. It remains to check what happens to the various applications of the cut rule (we can assume that the original proof doesn't use weakening). An application of the form $C \vee x, D \vee \bar{x} \vdash C \vee D$ transforms into an instance of the weakening rule: $C \vdash C \vee D$. If $y \neq x$, then the cut $C \vee y, D \vee \bar{y} \vdash C \vee D$ transforms into another instance of the cut rule, obtained by removing x from C, D; or disappears from the proof, if $\bar{x}$ appears in C or D.

We would now like to combine this with the previous observation

$$w(\phi \vdash \bot) \leq \max(w(\phi_{x=b} \vdash \bot) + 1, w(\phi_{x=\bar{b}} \vdash \bot), w(\phi)),$$

valid for $b \in \{0, 1\}$. We are interested in proving a result of the following form: if ϕ is a kCNF which can be refuted in tree-like Resolution in length at most S, then it can be refuted in width $f(S, k)$ (not necessarily in a tree-like fashion). Using S for the refutation length of ϕ and S_0, S_1 for the refutation lengths of $\phi|_{x=0}, \phi_{x=1}$, this gives

$$f(S, k) \leq \max(f(S_b, k) + 1, f(S_{\bar{b}}, k), k).$$

Since $S = S_0 + S_1 + 1$, we know that either $S_0 \leq \lfloor S/2 \rfloor$ or $S_1 \leq \lfloor S/2 \rfloor$. Either way, we get

$$f(S, k) \leq \max(f(\lfloor S/2 \rfloor, k) + 1, f(S - 1, k), k),$$

with an initial condition $f(1, k) = 0$. A straightforward induction shows that we can bound

$$f(S, k) \leq \log_2 S + k.$$

We conclude that if the refutation width of the kCNF ϕ is w, then every tree-like Resolution refutation of ϕ has at least 2^{w-k} lines. If ϕ is a Tseitin contradiction of a bounded-degree expander on n vertices, then the resulting lower bound is $2^{\Omega(n)}$. This should be compared to the trivial upper bound of $2^{O(n)}$, which follows from the proof of implicational completeness.

DAG-like Resolution In a general Resolution refutation, the trees T_0, T_1 could share vertices, and so it is no longer the case that either $S_0 \leq S/2$ or $S_1 \leq S/2$. However, there is one thing that we can say. Suppose that ϕ is a CNF over n variables. If x is a variable chosen at random and $b \in \{0, 1\}$ is chosen at random, then the substitution $x = b$ kills a clause C (makes it trivially true) with probability $w(C)/2n$. In particular, we can find x, b such that $\phi|_{x=b}$ requires at most $(1 - 1/2n)S$ lines to prove. If the refutation is minimal (no clause is unused) then x must appear in both polarities, and so $\phi|_{x=\bar{b}}$ requires at most $S - 1$ lines to prove. This gives the following recurrence for $g(S, k)$, which is a bound on the width required to refute a kCNF which can be refuted in Resolution in length at most S:

$$g(S, k) \leq \max(g(1 - 1/2n)S, k) + 1, g(S - 1, k), k),$$

with initial condition $g(1, k) = 0$. A straightforward induction shows that

$$g(S, k) \leq \log_{(1-1/2n)^{-1}} S + k \leq 2n \ln S + k.$$

Therefore if ϕ is a kCNF which has refutation width w, then every Resolution refutation of ϕ contains at least $\exp[(w - k)/2n]$ many lines. Since $w \leq n$, this bound is useless.

At this point, we need to find some slack in the argument. The culprit is that whereas the probability that a random substitution $x = b$ kills a clause C is $w(C)/2n$, all we could say about the size of refuting $\phi|_{x=b}$ is that it decreases by a factor of $1 - 1/2n$; this lower-bounds $w(C)/2n$ by $1/2n$, which could be very pessimistic in general. This suggests deciding on some threshold d, and only considering clauses of width d in the argument. Let $g_d(S, k)$ be a bound on the width required to refute a kCNF which has a Resolution refutation in which at most S clauses have width more than d. The same reasoning as before now gives

$$g_d(S, k) \leq \max(g_d(1 - d/2n)S, k) + 1, g_d(S, k), k),$$

with initial condition $g_d(0,k) = d$. In order to turn this into an actual recurrence, we need to introduce another parameter which decreases in the branch $x = \bar{b}$; this could be, for example, the number of variables or the total number of lines. After doing that, we can solve the recurrence much as before, obtaining

$$g_d(S,k) \leq \log_{(1-d/2n)^{-1}} S + d + k \leq \frac{2n}{d} \ln S + d + k.$$

The best choice for d is $d = \sqrt{2n \ln S}$, which identifies the first two terms. It leads to the bound

$$g_d(S,k) \leq 2\sqrt{2n \ln S} + k.$$

If ϕ is a kCNF which has refutation width w, then $w \leq 2\sqrt{2nS}+k$, and so $w-k \leq \sqrt{8n \ln S}$. Therefore every Resolution refutation of ϕ contains at least $\exp[(w-k)^2/8n]$ many lines. This is non-trivial if $w \gg \sqrt{n}+k$. For example, if ϕ is a Tseitin contradiction of a bounded-degree expander on n vertices then $k = O(1)$ and $w = \Omega(n)$, giving a bound of $2^{\Omega(n)}$.

5.6 Random kSAT

Tseitin contradictions are CNFs with a very particular structure which makes them hard to refute. Perhaps this is a fluke, and most unsatisfiable kCNFs are easy to refute? In order to address this question formally, we first need to explain how to sample a random kCNF. The most reasonable way is to fix a given number of variables n and a given number of clauses m, and sample m random clauses. There are two natural models of random clauses: one model simply samples k random literals, and the other samples k random literals *corresponding to different variables*. We will use the latter model, though for the values of m that we will consider, the two models are almost identical.

How many clauses do we need to sample so that a random kCNF is unsatisfiable? If α is any fixed truth assignment and C is a random clause, then the probability that α satisfies C is $1 - 2^{-k}$. Therefore α satisfies the CNF with probability $(1-2^{-k})^m$. If $(1-2^{-k})^m = o(2^n)$, then the expected number of satisfying assignments is $o(1)$, and so with high probability, the kCNF is unsatisfiable. This happens for $m \geq D_k n$, for an appropriate value of D_k.[6] We will attempt to prove that a random kCNF is hard to refute in Resolution, for the largest possible *clause density* m/n, where n is the number of variables and m is the number of clauses. We will do so by proving a width lower bound.

In order to prove a width lower bound on refuting a random kCNF ϕ with n variables and $m \geq D_k n$ clauses, we emulate the proof of the width lower bound for Tseitin contradictions. We start by defining a complexity measure μ for clauses: for a clause C, we define $\mu(C)$ as the minimum number of clauses of ϕ which imply C. If C is derived from C_1, C_2 by the cut rule, then $\mu(C) \leq \mu(C_1) + \mu(C_2)$, and so any refutation must contain a clause C such that $M/2 \leq \mu(C) \leq M$, where M is an arbitrary parameter satisfying $M \leq \mu(\bot)$.

The next step is to argue that C must be wide. Just as in the case of Tseitin contradictions, we argue that if S is a minimal set of clauses of ϕ which imply C, and x is a variable which is mentioned in a unique clause $D \in S$, then x must appear in ϕ. Indeed, since x appears only in D, the proof of C cannot use D, and so C already follows from $S \setminus \{D\}$. Therefore to complete the argument, we need to show that if $M/2 \leq |S| \leq M$ then there must be many variables which appear in only one clause in S.

We will now determine the largest number of clauses m (as a function of the number of variables n) for which we can prove a strong width lower bound for 3CNFs (we leave the case of general k for the reader). Our calculations will repeatedly use the following two estimates. First, the probability that a clause mentions only variables in some set of size t is

$$\frac{\binom{t}{3}}{\binom{n}{3}} \leq \left(\frac{t}{n}\right)^3.$$

[6] The optimal value of D was calculated by physicists using the cavity method [MPZ02, MMZ06]. It is known rigorously only for large k [DSS22], in which case it coincides with the value computed by the physicists.

Second, we will need the following standard estimate on binomial coefficients:

$$\binom{n}{t} \leq \left(\frac{en}{t}\right)^t.$$

Let us start with our first task: showing that $\mu(\bot)$ is large, that is, that any small number of clauses is satisfiable. The standard trick to show that a set S of clauses is satisfiable is to find a matching between the clauses in S and variables. That is, we will associate with each clause C_i in S a variable x_i, in such a way that different clauses are matched with different variables. We can use x_i to satisfy C_i, thus showing that the clauses in S are satisfiable. We show that a matching exists using Hall's theorem. This requires us show that any subset T of S mentions at least $|T|$ variables. If we aim at a bound of $\mu(\bot) > M$, then we need this to hold for all subsets T of size at most M.

The probability that a subset of size t mentions fewer than t variables is at most $\binom{n}{t-1}(\frac{t-1}{n})^{3t}$. Hence the probability that our 3CNF contains such a bad subset is at most

$$\binom{m}{t}\binom{n}{t}\left(\frac{t}{n}\right)^{3t} \leq \left(\frac{em}{t} \cdot \frac{en}{t} \cdot \frac{t^3}{n^3}\right)^t = \left(\frac{e^2 mt}{n^2}\right)^t.$$

We need this number to be small, which requires $mt \ll n^2$, that is, $t \ll n^2/m$. Roughly speaking, this shows that $\mu(\bot) \gg n^2/m$ with high probability.

Second, we need to show that every small set of clauses contains many variables which are mentioned exactly once, a phenomenon known as *unique expansion*. The more familiar condition is *expansion*, which requires every small set S of clauses to mention a number of variables proportional to its size, say at least $\lambda|S|$. How many of these *neighbors* of S must be unique? If there are $\kappa|S|$ unique neighbors then each of the other $(\lambda - \kappa)|S|$ neighbors appears at least twice. Since every clause mentions exactly three variables, this shows that $\kappa|S| + 2(\lambda - \kappa)|S| \leq 3|S|$, and so $\kappa \geq 2\lambda - 3$. In other words, in order to get unique expansion, we need expansion more than $3/2$.

The probability that there exists a subset of size t which mentions fewer than λt variables is at most

$$\binom{m}{t}\binom{n}{\lambda t}\left(\frac{\lambda t}{n}\right)^{3t} \leq \left(\frac{em}{t} \cdot \frac{e^\lambda n^\lambda}{\lambda^\lambda t^\lambda} \cdot \frac{\lambda^3 t^3}{n^3}\right)^t = \left(C_\lambda \frac{mt^{2-\lambda}}{n^{3-\lambda}}\right)^t,$$

where C_λ is some constant depending on λ; since $\lambda \leq 3$, we can replace it by a universal constant. For this to be small, we need $mt^{2-\lambda} \ll n^{3-\lambda}$.

We are aiming at some constant unique expansion κ, which requires us to choose $\lambda = 3/2 + \kappa/2$. Our width lower bound works for a parameter M whenever both parts of the argument work: every subset of size $t \leq M$ mentions at least t variables, and every subset of size $t \leq M$ mentions at least λt variables. The second condition is stronger, and requires $M^{2-\lambda} \ll n^{3-\lambda}/m$, that is, $M^{1/2-\kappa/2} \ll n^{3/2-\kappa/2}/m$. Since $(1/2-\kappa/2)^{-1} \approx 2(1+\kappa)$, this holds roughly when $M \ll n^{(3-\kappa)(1+\kappa)}/m^{2(1+\kappa)} \approx n^{3+2\kappa}/m^{2+2\kappa}$. The resulting width lower bound is $\kappa M/2$, which corresponds to a length lower bound of $\exp[(\kappa M/2-3)^2/8n]$. This lower bound is meaningful whenever $M^2 \gg n$, and it is exponential whenever $M^2 \gg n^{1+\epsilon}$ for any $\epsilon > 0$ (here by exponential we mean any bound of the form $2^{\Omega(n^\delta)}$).

Let us now be even rougher with our estimates, ignoring polynomial factors in n^κ. The maximum value of M we can use is n^3/m^2 (we obtain this from $n^{3+2\kappa}/m^{2+2\kappa}$ by neglecting κ). For this to be non-trivial we need $m \ll n^{3/2}$. In order to get an exponential lower bound we need $M^2 \gg n$, which translates to $n^6/m^4 \gg n$ and so $m \ll n^{5/4}$. Bringing back the factors which we removed, this technique works whenever the number of clauses is at most $n^{5/4-\epsilon}$, for any $\epsilon > 0$.

We can improve on this analysis by giving a better bound on the width of a clause in terms of its μ-complexity.[7] Suppose that S is a minimal set of clauses of ϕ which imply the clause C. Substitute variables for the variables in C so that C becomes false; the resulting set $S|_{C=0}$ is minimally unsatisfiable.

[7]This argument is taken from Eli Ben-Sasson's PhD thesis [BS01].

If all subsets $T \subseteq S|_{C=0}$ mentioned at least $|T|$ variables, then $S|_{C=0}$ would be satisfiable by Hall's theorem (we used this argument above to show that no small set of clauses is contradictory). Hence there must be some such subset T which mentions fewer than $|T|$ variables. If we choose T to be maximal, then by maximality, any subset $R \subseteq S|_{C=0} \setminus T$ must mention at least $|R|$ variables, and so $S|_{C=0} \setminus T$ is satisfiable. Thus T must be unsatisfiable. Since $S|_{C=0}$ was minimally unsatisfiable, it must be that $T = S|_{C=0}$, and so $S|_{C=0}$ mentions fewer than $|S|$ variables. We conclude that S mentions fewer than $|S| + w(C)$ variables.[8]

Now suppose that $\mu(C) \leq M$, and that every subset of at most M clauses has expansion at least λ, which happens (with high probability) as long as $M^{2-\lambda} \ll n^{3-\lambda}/m$. If S is a minimal set of clauses of ϕ implying C then on the one hand, S mentions at least $\lambda|S|$ variables, and on the other hand, it mentions at most $|S| + w(C)$ variables. The width of C is thus at least $(\lambda - 1)|S|$.

Our previous proof now works out as long as $\lambda \geq 1 + \kappa$ for some $\kappa > 0$, which is the analog of unique expansion (compare this with the earlier condition $\lambda \geq 3/2 + \kappa/2$). The condition on M is now $M^{1+\kappa} \ll n^{2+\kappa}/m$. Ignoring factors polynomial in n^κ, we can take $M \approx n^2/m$. In order to get an exponential lower bound we still need $M^2 \gg n$, which translates to $n^4/m^2 \gg n$ and so $m \ll n^{3/2}$. This is the best value of m currently known.

Finally, let us check what happens if instead of 3CNFs, we consider kCNFs. The probability that a subset of size t mentions fewer than λt variables is at most

$$\binom{m}{t}\binom{n}{\lambda t}\left(\frac{\lambda t}{n}\right)^{kt} \leq \left(\frac{em}{t} \cdot \frac{(en)^\lambda}{(\lambda t)^{\lambda t}} \cdot \frac{(\lambda t)^k}{n^k}\right)^t = \left(C_{\lambda,t}\frac{mt^{k-1-\lambda}}{n^{k-\lambda}}\right)^t.$$

We need this number to be small, and so $mt^{k-1-\lambda} \ll n^{k-\lambda}$. Roughly speaking, $\lambda \approx 1$, and so this works for t up to roughly $(n^{k-1}/m)^{1/(k-2)}$. This is our value of M. In order to get a meaningful bound we need $M^2 \gg n$, which translates to $(n^{k-1}/m)^2 \gg n^{k-2}$, and so $m \ll n^{k/2}$.

5.7 Certifying unsatisfiability

We have just seen that even though a random kCNF with $C_k n$ clauses is unsatisfiable with high probability, Resolution needs an exponential number of lines to refute a random kCNF even with roughly $n^{k/2}$ clauses. It is not known whether Resolution can refute random kCNFs with $n^{k/2}$ clauses. However, there are other proofs systems which are able to refute such kCNFs, notably TC^0-Frege (described below); and moreover, such proofs can be found efficiently.

Refuting random 4CNFs We start by describing an approach to refuting random 4CNFs due to Goerdt and Krivelevich [GK01]; the same approach works for refuting random kCNFs whenever k is even.

Given a random 4CNF φ with n variables and m clauses, the idea is to show that if y is a satisfying assignment then:

1. y contains fewer than $n/2$ many false variables.
2. y contains fewer than $n/2$ many true variables.

Together, this implies that no y can exist.

We know that the all-false assignment is almost certainly not a satisfying assignment since φ almost certainly contains a *positive clause*, that is, a clause of the form $x_i \vee x_j \vee x_k \vee x_\ell$. This suggests that positive clauses are an obstruction for y containing many false variables. To make this precise, consider the hypergraph H whose vertices are the variables, and whose hyperedges are the positive clauses. A set of vertices is an independent set in H if assigning these variables false does not falsifies any positive clause. Therefore, we can bound the number of false variables in any assignment y by $\alpha(H)$, the size of the largest independent set in H.

Bounding independent sets in hypergraphs is hard, but there are several known techniques for bounding the size of independent sets in graph. The most prominent such technique is due to Lovász [Lov79], and has

[8] This argument (with $C = \bot$) is attributed to Tarsi, see [AL86, Ref. 7].

found many applications in combinatorics, chiefly in Erdős–Ko–Rado theory and in coding theory, where it is behind the celebrated linear programming bound.[9] The idea is that if $G = (V, E)$ is a graph and A is a $V \times V$ symmetric matrix with $A_{uv} = 1$ whenever $(u, v) \notin E$, then $\lambda(A)$, which is the maximum eigenvalue of A, bounds $\alpha(G)$. Indeed, if S is an independent set of G then on the one hand,

$$1_S^T A 1_S = \sum_{u,v \in S} A_{uv} = \sum_{u,v \in S} 1 = |S|^2,$$

and on the other hand,

$$1_S^T A 1_S \leq \lambda(A) 1_S^T 1_S = \lambda(A)|S|.$$

While H is a hypergraph rather than a graph, there is a simple way to construct a graph G out of it. We take as vertices the set of all ordered pairs of variables, and connect (x_i, x_j) to (x_k, x_ℓ) if ϕ contains the clause $x_i \vee x_j \vee x_k \vee x_\ell$; for the sake of analysis, we insist that the variables appear in the clause in this particular order. If S is the set of variables assigned false in α then $S \times S$ is an independent set in G, and so $|S| \leq \sqrt{\alpha(G)}$.

The next step is to construct an appropriate matrix A. If $\{u, v\}$ is not an edge then we have to choose $A_{uv} = 1$; this includes the diagonal. What should we put in A if $\{u, v\}$ is an edge? In view of applying results from random matrix theory, we choose a value $-\alpha$ that results in $\mathbb{E}[A_{uv}] = 0$ for off-diagonal entries. Roughly speaking, G contains $m/16$ random edges, which is a p-fraction for $p \approx m/8n^4$ (since there are roughly $n^4/2$ potential edges). The value α should satisfy $1 - p = p\alpha$, and so $\alpha = (1 - p)/p$.

What can we say about the eigenvalues of A? To figure out, let us consider the trace of A^2. On the one hand, this is the sum of the squares of the eigenvalues of A. On the other hand, it is equal to the sum of squares of entries of A. The expected square of an off-diagonal entry A_{ij}, which is also its variance σ^2, is

$$\sigma^2 = \mathbb{E}[A_{ij}^2] = (1 - p) \cdot 1^2 + p \cdot \alpha^2 = 1 - p + \frac{(1 - p)^2}{p} = \frac{1 - p}{p}.$$

Therefore the sum of the squares of the eigenvalues of A is approximately $\frac{1-p}{p} n^4$ in expectation. Since there are n^2 such eigenvalues, we expect the eigenvalues of A to scale like $\sqrt{\frac{1-p}{p} n^4/n^2} = \sqrt{\frac{1-p}{p}} n$. The classical *semicircle law* implies (under some assumptions on p) that if we plot the density of the eigenvalues of A, then in the limit we get a semicircle whose endpoints are $\pm 2\sigma n$ (here n is the square root of the number of vertices). Füredi and Komlós [FK81] proved something stronger (again, under some assumptions on p): with high probability, there are *no* eigenvalues beyond the strip $\pm(1 + o(1))2\sigma n$.

Backing out, our work so far shows that with high probability, we can efficiently certify that $\alpha(G) \lesssim 2\sigma n$. The corresponding bound on the number of variables assigned false in y is $\sqrt{\alpha(G)}$. Since we want to show that fewer than half the variables are assigned false in y, we need $2\sigma n \ll n^2/4$, and so $\sigma^2 \ll (n/8)^2$. Substituting $\sigma^2 = \frac{1-p}{p}$ and recalling that $p = m/8n^4$, we need $\frac{n^4}{m} \ll n^2$ (ignoring constant factors) and so $m \gg n^2$.

Recapping, we have shown that if $m \gg n^2$ then we can show (with high probability) that every satisfying assignment of ϕ contains fewer than $n/2$ variables assigned false, since our spectral bound shows that $\alpha(G) < n^2/4$. Similarly, with high probability we can show that every satisfying assignment of ϕ contains fewer than $n/2$ variables assigned true, and so no satisfying assignment exists.

Our certification procedure requires us to bound the maximum eigenvalue of the matrix A; in fact, it suffices to bound its spectral radius, which is the maximum magnitude of an eigenvalue. This is a standard problem in numerical analysis, with several practical solutions, typically based on the power method. Since the numbers involved are not necessarily rational, when implementing these procedures, computers use floating-point arithmetic, which involves certain approximations. Using techniques from numerical analysis, we can carefully keep track of the resulting errors, and show that polynomial precision suffices to obtain a rigorous bound on the spectral radius, which is good enough for our purposes.

[9] In Erdős–Ko–Rado theory, it is common to name this bound after Hoffman [Hae21]. The linear programming bound is due to Delsarte [Del73]. Its connection to the Lovász bound was pointed out by Schrijver [Sch79].

The transcript of an algorithm which bounds the maximum eigenvalue of A (as well as the maximum eigenvalue of the corresponding matrix for negative clauses) thus provides a *certificate of unsatisfiability*, which can be checked by an efficient algorithm.

It seems unlikely that we can convert such a certificate into a Resolution proof. Müller and Tzameret [MTz14] showed how to translate this kind of certificate into a proof in the stronger system TC^0-Frege, which is still quite weak. Whereas proofs in Resolution consist of a sequence of clauses, proofs in TC^0-Frege consist of a sequence of TC^0 circuits, that is, bounded depth circuits with threshold gates, and have derivation rules that correspond to the semantics of the gates. To certify that A has a small maximum eigenvalue, Müller and Tzameret use an approximate eigendecomposition of A, using a polynomial number of bits of precision.[10]

What happens for kCNFs for other values of k? If k is even, then we can convert the k-uniform hypergraph H into a graph G on $n^{k/2}$ vertices. If there are m clauses then the density is $p \approx m/n^k$ and so $\sigma^2 \approx 1/p \approx n^k/m$. Since A has $n^{k/2}$ many rows and columns, the bound on its maximum eigenvalue is of order $\sqrt{\sigma^2 n^{k/2}} \approx n^{3k/4}/\sqrt{m}$. This needs to be at most $(n/2)^{k/2}$, that is $n^{3k/4}/\sqrt{m} \ll n^{k/2}$, and so $m \gg n^{k/2}$.

Adaptation to random 3CNFs When k is odd, we cannot split each clause into two parts of width $k/2$. We could split the clauses into two unequal parts, but then instead of a graph we get a bipartite graph. Friedman and Goerdt [FG01, FGK05], whose approach was later improved by Goerdt and Lanka [GL03], suggested rectifying this using the following observation, which we illustrate in the specific case of $k = 3$. If we started with roughly $n^{3/2}$ clauses, then we expect to have roughly $(n^{3/2}/8)^2/n = n^2/64$ pairs of clauses of the form

$$(x_i \vee x_j \vee z), (x_k \vee x_\ell \vee \bar{z}).$$

We can resolve the pairs to obtain about $n^2/64$ clauses of the form $x_i \vee x_j \vee x_k \vee x_\ell$, thus reducing to the situation of random 4SAT (in practice, we need somewhat more than $n^2/64$ clauses, so we need to start with somewhat more than $n^{3/2}$ clauses).

Unfortunately, the resulting clauses do not have the correct distribution. For each variable z, we expect to have roughly $n^{1/2}/8$ clauses of the type $x_i \vee x_j \vee z$ and a similar number of clauses the type $x_k \vee x_\ell \vee \bar{z}$. The n resolvents that we get are some sort of Cartesian product of these two lists. In order to mix them better, when constructing the graph G, we assign pairs to edges in an alternate fashion:

$$(x_i \vee x_j \vee z), (x_k \vee x_\ell \vee \bar{z}) \mapsto \{(x_i, x_k), (x_j, x_\ell)\}.$$

(In order to get an undirected graph, we need to allow z to be a negative literal.) It turns out that the Füredi–Komlós bound still holds, though it needs to be reproved to accommodate this distribution.

A different approach Feige and Ofek [FO07] developed an altogether different approach for certifying random 3SAT. They observed that if an assignment y satisfies a clause $\ell_1 \vee \ell_2 \vee \ell_3$, then either $\ell_1 \oplus \ell_2 \oplus \ell_3 = 1$ or ℓ_1, ℓ_2, ℓ_3 are not all equal (or both). Now a random assignment satisfies about $1/2$ the clauses in the first sense, and about $3/4$ of the clauses in the second sense. Furthermore, it satisfies about half the literals in each clause.

Suppose for the moment that this was the case for *all* assignments (which happens when there are enough clauses). Given a truth assignment, if the fraction of clauses with $i \in \{0, 1, 2, 3\}$ satisfied literals is c_i, then $c_1 + c_3 \lesssim 1/2$, $c_1 + c_2 \lesssim 3/4$, and $c_1 + 2c_2 + 3c_3 \approx 3/2$. This allows us to bound the fraction of satisfied clauses $c_1 + c_2 + c_3$, by considering the correct linear combination of the above inequalities:

$$c_1 + c_2 + c_3 = \frac{c_1 + c_3}{4} + \frac{c_1 + c_2}{2} + \frac{c_1 + 2c_2 + 3c_3}{4} \lesssim \frac{7}{8},$$

as expected.

This reduces the certification task to three easier certification tasks:

[10] Müller and Tzameret do not attempt to show that this decomposition can be found in polynomial time, but this seems likely.

1. Show that every assignment satisfies $c_1 + c_3 \lesssim 1/2$.
2. Show that every assignment satisfies $c_1 + c_2 \lesssim 3/4$.
3. Show that every assignment satisfies $c_1 + 2c_3 + 3c_3 \lesssim 3/2$.

There is slack in our calculations, and we use it to simplify our first task: we will only prove that $c_1 + c_3 \lesssim 1$.

No assignment satisfies almost all clauses as XOR The idea is to reduce this to a similar certification task on *2XOR* (compare this to the reduction from 3CNF to 4CNF considered above). Given two clauses $\ell_{i'} \oplus \ell_j \oplus \ell_k = 1$ and $\ell_{i''} \oplus \ell_j \oplus \ell_k = 1$, we can deduce the clause $\ell_{i'} \oplus \ell_{i''} = 0$, that is, $\ell_{i'} = \ell_{i''}$. If we start with m clauses, then we end up with $m^2/4n^2$ clauses.

If the original instance were almost satisfiable as 3XOR, then the new instance would be almost satisfiable. This corresponds to a cut containing almost all edges in the graph whose vertices are the $2n$ literals, and in which each clause $\ell_{i'} = \ell_{i''}$ corresponds to an edge $(\ell_{i'}, \neg\ell_{i''})$. Altogether, there are $m^2/4n^2$ edges. A random graph on cn edges is almost bipartite for small c, but if $m^2/4n^2 \geq Cn$ for large C then it is far enough from being bipartite for our purposes. Moreover, this can be certified using a simple spectral argument, which capitalizes on the observation that the number of edges cut by a partition $(S, \bar{S})$ can be related to $1_S^T A 1_S$, where A is the adjacency matrix.

For this argument to work, we need $m^2/n^2 \gg n$, and so $m \gg n^{3/2}$.

No assignment satisfies significantly more than $3/4$ of all clauses as NAE Consider a constraint $\mathsf{NAE}(\ell_1, \ell_2, \ell_3)$, stating that ℓ_1, ℓ_2, ℓ_3 are not all equal. If we count the number of disagreements among pairs of literals, then the constraint is satisfied if the number is 2, and not satisfied if the number is 0. This allows us to convert the NAE version of ϕ to an instance ψ of maximum cut, related as follows: $\frac{2}{3}v(\phi) = v(\psi)$, where $v(\phi)$ is the maximum fraction of clauses of ϕ satisfied as not-all-equal, and $v(\psi)$ is the maximum fraction of edges which can be cut in ψ. Since ψ is a dense instance (containing $3m$ edges on n vertices, where $m \ggg n$), we expect $v(\psi) \approx 1/2$, and this can be certified spectrally. In turn, this implies that $v(\phi) \approx \frac{3}{4}$.

Every assignment satisfies an average of $3/2$ literals per clause In a random 3CNF, each variables appears roughly the same number of times positively and negatively, and so each assignment satisfies roughly half of its appearances. It follows that every assignment satisfies roughly half the literals in all clauses put together.

Feige–Kim–Ofek witnesses The argument of Feige and Ofek gets stuck at $n^{3/2}$ clauses due to the way in which they show that $c_1 + c_2 \lesssim 1$. Feige, Kim and Ofek [**?**] introduce a combinatorial certificate based on parity, and use this to certify random 3CNFs with only $n^{1.4}$ clauses. However, their combinatorial certificate, while easy to verify, is hard to find. It thus constitutes a "nondeterministic" certification algorithm.

Müller and Tzameret showed that Feige–Kim–Ofek certificates can also be converted into TC^0-Frege proofs, concluding that TC^0-Frege is stronger than Resolution for refuting random 3CNFs. This also explains why nondeterministic certification algorithms are interesting: they correspond to proofs. The gap between certification algorithms and nondeterministic certification algorithms is similar to the gap between SAT solvers and the Resolution proof system.

6 Exponential time algorithms for kSAT

Given a kCNF on n variables, we can always determine whether it is satisfiable by going over all 2^n truth assignments. Can we do better? The *exponential time hypothesis (ETH)* states that no algorithm for kSAT can run in $2^{o(n)}$ time. The exponential time hypothesis can be used to prove many other hardness results. Roughly speaking, to show a lower bound on a problem P, we reduce kSAT to P, and deduce that a too-good algorithm for P would refute ETH. The reduction often takes exponential time, and so the exponential lower bound on kSAT implies a polynomial lower bound on P.

The exponential time hypothesis is qualitative in nature. It leaves a gap between $2^{o(n)}$, which is ruled out, to the trivial 2^n algorithm. It is natural to conjecture a 2^n lower bound, but it turns out that for every constant k, we can actually improve on this exponentially. We describe two such algorithms in this section, one due to Schöning [Sch02], the other due to Paturi, Pudlák and Zane [PPZ99], and known as the PPZ algorithm.

The *strong exponential time hypothesis (SETH)* is the next best thing: it states that for every $\delta > 0$ we can find some constant k such that solving kSAT requires time $2^{(1-\delta)n}$. This hypothesis is important in the recently emerging area of *hardness in P* (also known as *fine-grained complexity*), which aims at proving sharp lower bounds for problems in P.

All algorithms we present work in time $O^*((2 - \Omega(1/k))^n)$ on kCNFs (the O^* notation hides factors polynomial in n and in the input length). Vyas and Williams [VW19] showed that the PPZ algorithm works in time $O^*((2 - \Omega(\log k/k))^n)$ on *random* kCNFs whose clause density is close to the satisfiability threshold. Lincoln and Yedidia [LY20] showed that a modification of Schöning's algorithm works in time $O^*((2 - \Omega(\log^2 k/k))^n)$ in the same setting.

6.1 Schöning's algorithm

The first algorithm we present is due to Schöning [Sch02]. For concreteness, we will first describe Schöning's algorithm in the case of 3SAT, though the analysis for general k is identical. Schöning's algorithm assumes that the given 3CNF is satisfiable, and shows how to find a satisfying assignment with high probability in time α^n, for some $\alpha < 2$. If the 3CNF were unsatisfiable, then after α^n steps the algorithm would not find a satisfying assignment, and so we can conclude that the 3CNF is probably unsatisfiable.

Suppose that we are given a 3CNF ϕ and an assignment x. If x is satisfying, then we are done: ϕ is satisfiable. Otherwise, we can try to improve x. One natural idea is to choose a clause C which x falsifies, and flip one of its variables. This will satisfy C, but might falsify other clauses.

To track our progress, we will measure our distance to a particular satisfying assignment x^*. Since x falsifies C while x^* satisfies C, the two assignments disagree on at least one of the variables in C. With probability $p \geq 1/3$, we flip one of the variables of C on which x and x^* disagree, thus decreasing the distance between x and x^* by 1. With probability $1 - p$, we flip a variable on which x and x^* agree, and so their distance increases by 1.

In the worst case, $p = 1/3$, and so on average, x gets farther away from x^*. Nevertheless, we will see that there is some small but non-negligible probability that a few iterations of this process succeed in "correcting" x to the assignment x^*.

Let d be the initial distance between x and x^*; if we choose x at random, then $d \sim \mathrm{Bin}(n, 1/2)$. If we run the random correction process outlined above for d or more steps (n steps suffice), then it corrects x to x^* with probability at least $(1/3)^d$. Therefore the algorithm succeeds with probability at least

$$\mathop{\mathbb{E}}_{d\sim\mathrm{Bin}(n,1/2)}\left[\frac{1}{3^d}\right] = \left(\mathop{\mathbb{E}}_{b\sim\mathrm{Ber}(1/2)}\left[\frac{1}{3^b}\right]\right)^n = \left(\frac{1}{2}\cdot 1 + \frac{1}{2}\cdot\frac{1}{3}\right)^n = \left(\frac{2}{3}\right)^n.$$

This analysis insists that we always choose a correct variable. We can obtain a better estimate by allowing the algorithm to make wrong choices. We can model the distance from x to x^* as a random walk. The starting point is the initial distance d between x and x^*, which has distribution $d \sim \mathrm{Bin}(n, 1/2)$. Assuming the worst case (an assumption which we justify below), at every step the distance decreases by 1 with probability 1/3,

and increases by 1 with probability 2/3. Ignoring the fact that the distance can be at most n (which is in our favor — as the distance gets very close to n, we obtain better bounds on p), this is just a biased random walk on the line. Denoting by p_i the probability that the walk ever reaches the origin if it starts at i, we get the recurrence

$$p_i = \frac{1}{3}p_{i-1} + \frac{2}{3}p_{i+1}, \quad p_0 = 1, \quad \lim_{i\to\infty} p_i = 0.$$

This is a homogeneous recurrence relation with constant coefficients. The roots of the characteristic polynomial $(2/3)x^2 - x + (1/3)$ are easily calculated: $1, 1/2$. Therefore $p_i = A(1/2)^i + B$. Using the two initial conditions, we easily determine that $B = 0$ (due to the limit at infinity) and that $A = 1$ (due to the value of p_0), and so $p_i = 1/2^i$.

How long does it take the random walk to reach the origin? After (say) Cn steps, where C is a large enough constant, the walk is expected to be at position at least $(2/3 - 1/3)Cn = (C/3)n$, and so it is very likely to be at position (say) at least $(C/4)n$; in fact, this will continue to hold from this point on, and combining tail bounds with the union bound (details appear below), we can conclude that if the random walk ever reaches the origin, then it is very likely to happen within the first Cn steps. This suggests the following procedure:

1. Choose an initial assignment x at random.

2. Run the random correction process for Cn steps.

If the random correction process succeeds in finding a satisfying assignment, then ϕ is satisfiable. If the initial distance between x and x^* is d, then this happens with probability at least roughly $1/2^d$ (this is only a lower bound, since the process could end at a satisfying assignment which is different from x^*). Since $d \sim \mathrm{Bin}(n, 1/2)$, the procedure succeeds with probability

$$\mathop{\mathbb{E}}_{d\sim\mathrm{Bin}(n,1/2)}\left[\frac{1}{2^d}\right] = \left(\mathop{\mathbb{E}}_{b\sim\mathrm{Ber}(1/2)}\left[\frac{1}{2^b}\right]\right)^n = \left(\frac{1}{2}\cdot 1 + \frac{1}{2}\cdot\frac{1}{2}\right)^n = \left(\frac{3}{4}\right)^n.$$

Therefore, if ϕ is satisfiable and we repeat this procedure about $(4/3)^n$ many times, it is likely to find a satisfying assignment.

The final algorithm thus repeats the procedure about $(4/3)^n$ many times. If one of the runs ends at a satisfying assignment, we know that ϕ is satisfiable, and otherwise, we guess that it is not satisfiable. Choosing parameters carefully, we can ensure that the answer is correct with high probability. Since the procedure runs in polynomial time, the overall running time is $O^*((4/3)^n)$ (the notation O^* hides polynomial factors in n).

If the input is a kCNF rather than a 3CNF, then the only thing that changes is the lower bound on the probability p: now it is only $1/k$. The characteristic polynomial $(1-1/k)x^2 - x + 1/k$ now has the two roots $1, 1/(k-1)$, and so $p_i = 1/(k-1)^i$. The success probability of the procedure is now

$$\left(\frac{1}{2}\cdot 1 + \frac{1}{2}\cdot\frac{1}{k-1}\right)^n = \left(\frac{k}{2(k-1)}\right)^n,$$

and so the running time is $O^*((2-2/k)^n)$.

More details: coupling Above we have assumed that $p = 1/3$ always, an assumption which isn't justified. For example, if x is the complement of x^*, then $p = 1$. Here is why the argument is valid nevertheless. Let d_t be the distance between x and x^* after t steps (we stop at $d_T = 0$), and let p_t be the probability that $d_{t+1} = d_t - 1$; thus $p_t \geq 1/3$.

Given d_0, we can sample $d_0, \ldots, d_T$ together with an infinite sequence $e_0, e_1, \ldots$ as follows. We start with $e_0 = d_0$. If $d_t > 0$, then we sample d_{t+1} and e_{t+1} in a coupled way, as follows:

1. With probability 1/3: let $d_{t+1} = d_t - 1$ and $e_{t+1} = e_t - 1$.

2. With probability $p_t - 1/3$: let $d_{t+1} = d_t - 1$ and $e_{t+1} = e_t + 1$.

3. With probability $1 - p_t$: let $d_{t+1} = d_t + 1$ and $e_{t+1} = e_t + 1$.

We sample e_{T+1} and beyond using the rule $e_{t+1} = e_t - 1$ with probability $1/3$ and $e_{t+1} = e_t + 1$ with probability $2/3$.

By construction, $d_t \leq e_t$ for all $t \leq T$, and $(e_t)_0^\infty$ is precisely the random walk with constant bias $p = 1/3$ considered above. If $e_t = 0$ then $d_s = 0$ for some $s \leq t$, and in particular, if the probability that e_t hits zero within Cn steps is (roughly) $(3/4)^n$, then the probability that d_t hits zero within Cn steps is *at least* (roughly) $(3/4)^n$.

More details: number of steps Above we claimed that it is extremely unlikely that the biased random walk hits zero after Cn steps or more, for large enough C. To show this, we need to use a celebrated large deviation bound known as the *Chernoff bound*.

If the walk starts at d, then after T steps, it is found at position

$$d + T - 2\mathrm{Bin}(T, 1/3) \geq T - 2\mathrm{Bin}(T, 1/3).$$

Let X be the value of the binomial random variable $\mathrm{Bin}(T, 1/3)$ appearing in the above expression. If $X < T/2$ then the random walk does not hit zero at step T. Chernoff's bound, in one form, states that

$$\Pr[X \geq T/2] = \Pr[X \geq \tfrac{3}{2}\,\mathbb{E}[X]] \leq e^{-\mathbb{E}[X]/10} = e^{-T/30}.$$

Applying the union bound, we see that the random walk hits zero at some step $T \geq Cn$ with probability at most

$$\sum_{T=Cn}^{\infty} e^{-T/30} = \frac{e^{-Cn/30}}{1 - e^{-1/30}} \leq 31e^{-Cn/30}.$$

Choosing $C = 30$, this probability is much smaller than 2^{-n}, and so the probability that the random walk hits zero within the first Cn steps is at least $2^{-d} - 2^{-n} \geq 2^{-d}/2$, which suffices for the analysis (instead of $(3/4)^n$ we get $(3/4)^n/2$).

The same argument also shows that $\lim_{i\to\infty} p_i = 0$. In fact, it shows that $p_T \leq 31e^{-T/30}$, since a random walk starting at T cannot hit zero before step T.

Derandomization Dantsin, Goerdt, Hirsch, Kannan, Kleinberg, Papadimitriou, Raghavan and Schöning [DGH+02] showed how to derandomize Schöning's algorithm, albeit with a loss in the running time. This is important since the ETH and SETH conjectures are about deterministic algorithms. For concreteness, we describe the derandomization for 3CNFs, though the general case is identical.

Schöning's algorithm has two random components: the random choice of x, and the random choices involved in the random correction process, namely the choice of which variable to flip in the chosen clause (the clause can be chosen deterministically, say the first falsified clause). The starting point is the observation that we can get rid of the randomness in the second step, if we only aim at a success probability of 3^{-d} rather than 2^{-d}, where d is the distance between x and some fixed satisfying assignment x^*.

Recall that 3^{-d} is a lower bound on the probability that during the the random correction process, we always choose a variable at which x and x^* disagree, in which case the process ends after exactly d steps. If we know that x and x^* are at distance at most d, this allows us to find x^* given x in time $O^*(3^d)$. To do this, we simulate d iterations of the random correction procedure, enumerating over all 3^d possible choices of randomness.

This suggests the following strategy. We zero in on a value of d such that there is a small set C which contains a point at distance at most d from every truth assignment. For each $x \in C$, we run the derandomized correction procedure. The overall running time is $O^*(|C| \cdot 3^d)$.

A set C which contains a point at distance at most d from every point in $\{0,1\}^n$ is known as a *covering code* of radius d. Every $x \in C$ is at distance at most d from $\binom{n}{\leq d}$ points in $\{0,1\}^n$, and so $|C| \geq 2^n / \binom{n}{\leq d}$

(this is the *sphere packing bound*). Conversely, there are covering codes which almost attain this bound (up to a $poly(n)$ factor), and these can be found in time $O^*(|C|)$ (here O^* hides $poly(n)$ factors).

It remains to choose a value of d which minimizes the running time $O^*(|C| \cdot 3^d)$. Up to $poly(n)$, factors,

$$\frac{1}{|C| \cdot 3^d} \gtrsim \frac{\binom{n}{d} 3^{-d}}{2^n}.$$

If we sum the right-hand side over all d, we obtain

$$\mathop{\mathbb{E}}_{d \sim \mathrm{Bin}(n,1/2)}[3^{-d}] = (3/4)^n.$$

In particular, there is a value of d such that

$$\frac{\binom{n}{d} 3^{-d}}{2^n} \geq \frac{(3/4)^n}{n+1}.$$

Moreover, $d \leq n/2$, since otherwise the left-hand side is at most $3^{-n/2}$, which is a lot smaller than $(3/4)^n$. Therefore $\binom{n}{d}$ and $\binom{n}{\leq d}$ are the same up to $poly(n)$ factors, and so for this value of d, the running time is $O^*((4/3)^n)$.

We can find the optimal value of d explicitly using the following well-known approximation for the binomial coefficient:

$$\binom{n}{\alpha n} \approx 2^{(1-h(\alpha))n},$$

where $h(\alpha) = -\alpha \log_2 \alpha - (1-\alpha)\log_2(1-\alpha)$ is the binary entropy function. If $d = \alpha n$ then the running time of the algorithm is roughly

$$\frac{2^n}{\binom{n}{\alpha n}} 3^{\alpha n} \approx 2^{h(\alpha)n} 3^{\alpha n}.$$

To find the optimal setting, define $g(\alpha) = 1 - h(\alpha) + \alpha \log_2 3$, and minimize g. Straightforward calculation gives $g'(\alpha) = \log_2 \frac{1-\alpha}{\alpha} + \log_2 3$, and so we need $\frac{1-\alpha}{\alpha} = 3$, resulting in $\alpha = 1/4$. We then have $2^{g(1/4)} = 3/2$, resulting in a running time of $O^*((3/2)^n)$, compared to $O^*((4/3)^n)$ for the randomized algorithm.

If the input is a kCNF rather than a 3CNF, then instead of $3^{\alpha n}$ we have $k^{\alpha n}$. The function g is replaced by $g_k(\alpha) = 1 - h(\alpha) + \alpha \log_2 k$, which is minimized when $\frac{1-\alpha}{\alpha} = k$, resulting in $\alpha = 1/(k+1)$. We then have

$$2^{g_k(1/(k+1))} = 2 \cdot \left(\frac{1}{k+1}\right)^{1/(k+1)} \left(\frac{k}{k+1}\right)^{k/(k+1)} \cdot k^{1/(k+1)} = \frac{2}{k+1} \cdot k^{k/(k+1)} \cdot k^{1/(k+1)} = \frac{2k}{k+1}.$$

Therefore the running time is $O^*((2 - 2/(k+1))^n)$, compared to $O^*((2-2/k)^n)$ in the randomized case.

Moser and Scheder [MS11] gave an improved derandomization which results in a $O^*((2 - 2/k + \epsilon)^n)$ algorithm for every $\epsilon > 0$. Roughly speaking, they were able to run the random correction procedure for more steps using a k-ary covering code to reduce the number of variable choices which need to be considered.

6.2 PPZ

Paturi, Pudlák and Zane [PPZ99] came up with a different idea for a kSAT algorithm. While the resulting algorithm is not better than Schöning's, further improvements of it are. Later on we will briefly describe one such improvement, the PPSZ algorithm [PPSZ05] (adding Saks to the list of authors).

Let us now proceed to describe the PPZ algorithm. Instead of describing a satisfiability algorithm, we will describe an algorithm which, given a satisfiable kCNF, finds a satisfying assignment with high probability. Such an algorithm can be turned into a satisfiability algorithm, just as we did for Schöning's algorithm.

Intuitively, the hardest satisfiable instances of kSAT are those in which there is a unique satisfying assignment x^*; these are the instances where the analysis of Schöning's algorithm is the tightest. (The

isolation lemma, discussed in Section 7, makes this intuition precise, by reducing SAT to the case in which the satisfying assignment is unique.) This means that if we flip any of the variables in x^* we must falsify some clause. In other words, for every variable x_i there is a clause C_i mentioning either x_i or $\bar{x}_i$, in which the x_i literal is the unique literal satisfying C_i in x^*.

Imagine now an exhaustive search procedure which goes over the variables $x_1, \ldots, x_n$ in some order. That is, for each x_i, we try both truth values in a recursive fashion. Consider the branch in which we guessed the values of all variables correctly (according to x^*). Suppose that among the variables appearing in C_i, the variable x_i is reached last. In this case, when reaching C_i, all other literals in C_i are false, and so the value of x_i is forced; this happens with probability $1/k$ over the random order. We expect to save about n/k guesses in this way. This suggests the following algorithm, whose input is a kCNF ϕ:

1. Choose a random assignment $y_1, \ldots, y_n \in \{0,1\}^n$.
2. Go over the variables $x_1, \ldots, x_n$ in a random order. For each x_i:
   (a) If x_i is the only literal in some clause of ϕ, set it in the way that satisfies the clause.
   (b) Otherwise, set x_i to y_i.
   (c) Simplify ϕ by substituting the value of x_i. For example, if x_i is set to true, then remove all clauses containing x_i, and remove $\bar{x}_i$ from all clauses containing this literal.
   (d) If one of the clauses becomes empty, give up.
3. Output the current assignment.

If we reach the last step, then we output a satisfying assignment. What is the probability that this happens, that is, that we always set x_i to x_i^*? For each x_j, if x_j is the last variable of C_j in the random order, and $x_k = x_k^*$ for all previously set variables x_k, then we always set x_j correctly, since its value is forced. Otherwise, we set x_j correctly with probability $1/2$. Hence, denoting by N the number of variables x_i which appear last out of all variables in C_i in the random order, the probability that the algorithm outputs a satisfying assignment is

$$\mathbb{E}[2^{-(n-N)}] \geq 2^{-(n-\mathbb{E}[N])} = 2^{-(n-k/n)} = 2^{-(1-1/k)n},$$

using Jensen's inequality in the first inequality. This means that we need to repeat the algorithm $2^{(1-1/k)n}$ times, resulting in a running time of $O^*((2^{1-1/k})^n)$. When $k = 3$, we have $2^{1-1/k} \approx 1.587 > 4/3$, so this is worse than Schöning's algorithm. What happens for other k? For PPZ to beat Schöning, we need $2 - 2/k > 2^{1-1/k}$, and so $1 - 1/k > 2^{-1/k}$, hence $(1-1/k)^k > 1/2$, which contradicts the well-known bound $(1-1/k)^k < 1/e$. Thus Schöning is always better.

Non-unique case Before describing how to improve the performance of the algorithm, let us see what happens when there is more than one satisfying assignment. In the case of a unique satisfying assignment, for every variable x_i there was a clause C_i in which the x_i-literal was the unique satisfying literal. This is not necessarily the case in general: for example, x_i could not appear in the CNF at all. What we can say for sure is that if x^* is *sensitive* to flipping x_i, that is, if flipping x_i results in an assignment which is not satisfying, then such a clause C_i does exist. Denoting by $s(x^*)$ the number of sensitive variables (known as the *sensitivity*), the same analysis as before shows that PPZ recovers the assignment x^* with probability

$$\mathbb{E}[2^{-(n-N)}] \geq 2^{-(n-\mathbb{E}[N])} = 2^{-(n-s(x^*)/k)}.$$

Therefore the success probability is

$$2^{-n} \sum_{x^*} (2^{1/k})^{s(x^*)}.$$

where the sum goes over all satisfying assignments.

It turns out that we can bound the sum inductively. For an arbitrary Boolean function f, let

$$S(f) = \sum_x f(x)(2^{1/k})^{s(f,x)},$$

where $s(f,x)$ is the number of coordinates i such that $f(x^{\oplus i}) \neq f(x)$. Let $S(m)$ be the minimum value of $S(f)$ over all $f\colon \{0,1\}^m \to \{0,1\}$ which are not identically zero; in our case, this nontriviality assumption corresponds to the input kCNF being satisfiable. Clearly $S(f) \geq S(f|_{x_m=0}) + S(f|_{x_m=1})$. However, this does not mean that $S(m) \geq 2S(m-1)$, since one of the functions $f|_{x_m=0}, f|_{x_m=1}$ could be identically zero. If $f|_{x_m=b}$ is identically zero then $S(f) = 2^{1/k} S(f|_{x_m=\bar{b}})$, since all satisfying assignments of $f|_{x_m=\bar{b}}$ are sensitive in the direction of x_m. Therefore

$$S(m) \geq \min(2S(m-1), 2^{1/k} S(m-1)) = 2^{1/k} S(m-1).$$

Since $S(1) = \min(1+1, 2^{1/k}) = 2^{1/k}$, this shows that $S(m) \geq 2^{m/k}$, which is tight for f with a unique satisfying assignment. Therefore the success probability of PPZ is always at least

$$2^{-n} S(n) \geq 2^{-n} 2^{n/k} = 2^{(1-1/k)n},$$

matching exactly what we got in the case of a unique satisfying assignment.

PPSZ The PPSZ algorithm [PPSZ05] improves on PPZ by making it easier for variables to be forced. In PPZ, a variable x_i is forced if it appears in a unit clause (a clause containing a single literal). In PPSZ, we run Resolution up to width w, trying to see if it implies a value for x_i. If $w = o(n/\log n)$, then due to automability of bounded width Resolution (see Section 5.3), the procedure for checking whether x_i is forced runs in subexponential time $2^{o(n)}$, which is "free" in this context (it actually suffices to take any $w = \omega(1)$ for the analysis to go through). This improves the probability that x_i is forced from $1/k$ to roughly

$$\frac{1}{k-1} \sum_{j=1}^{\infty} \frac{1}{j(j + \frac{1}{k-1})}.$$

For example, when $k = 3$, we improve $1/3$ to $2 - \ln 4 \approx 0.613706$, resulting in a running time of $O^*(2^{(\ln 4 - 1)n}) \approx 1.307^n$, which is better than Schöning's $(4/3)^n$. In fact, the original paper only proved this bound in the case of a unique satisfying assignment, and only later on Hertli [Her14] extended the analysis to the general case. The analysis was further improved by Scheder [Sch21]. Other authors have slightly improved on both Schöning and PPSZ (see Scheder's paper for pointers to the literature).

7 Isolation lemma and approximate counting/sampling

The worst case for both algorithms described in Section 6 is when the input CNF has a unique satisfying assignment, leading to the conjecture that such instances are worst-case. The corresponding promise problem is Unique-SAT, in which we have to tell apart unsatisfiable CNFs from ones with exactly one satisfying assignment.

In Section 1.4 we showed how to find a satisfying assignment using a satisfiability oracle, a process known as a search-to-decision reduction. The process is sequential in nature: we first figure out the value of x_1, then the value of x_2, and so on. We can parallelize the process when the satisfying assignment is unique: for each variable x_i in parallel, we can determine its value by fixing x_i to 0 and asking a satisfiability oracle whether this results in a satisfiable instance or not. This approach only works when the solution is unique: for example, the CNF corresponding to $x_1 \oplus x_2$ has solutions with $x_1 = 0$ or $x_2 = 0$, but not both.

Both of these applications motivate the isolation lemma, due to Valiant and Vazirani [VV86], which gives a reduction from SAT to Unique-SAT. The same ideas can also be used to approximately count the number of satisfying assignments, which can in turn be used to sample an approximately uniform satisfying assignment.

A different type of isolation lemma was used by Mulmuley, Vazirani and Vazirani [MVV87] to give a parallel algorithm for perfect matching. The idea there is to add random weights in order to make the minimum weight perfect matching unique. Noam Ta-Shma [Ta-15] gave a very short proof of this isolation lemma.

The notes below incorporate material from lecture notes of Ryan O'Donnell and (separately) Dana Moshkovitz.

7.1 Unique-SAT and the isolation lemma

Suppose that ϕ is a satisfiable CNF on n variables with M satisfying assignment. In order to reduce the number of satisfying assignments, we need to cut down the number of satisfying assignments by a factor of M. We can do so using a good hash function $h\colon \{0,1\}^n \to \{1,\ldots,M\}$ by asking for a satisfying assignment x which hashes to some random value. In order to incorporate h into the CNF, we need h to be efficiently computable.

If h is a uniform hash function, meaning that given a random assignment, the output is uniformly random, then M must be a power of 2. Therefore we slightly adjust our idea, using a hash function whose range is $K \approx M$, where K is a power of 2.

If h is a uniformly random function and we choose $H \in \{1,\ldots,K\}$ at random, then the probability that $\phi \wedge$ "$h(x) = H$" has a unique satisfying assignment is

$$\frac{M}{K}\left(1 - \frac{1}{K}\right)^{M-1},$$

suggesting that we choose K to be the smallest power of 2 exceeding M.

Uniformly random hash functions cannot be computed efficiently, and so we cannot use them. Instead, we have to make do with a pseudorandom function. The minimal assumption is that for every x, $\Pr_{h,H}[h(x) = H] = 1/K$, but this is not enough: a random constant function satisfies this constraint. The next relevant assumption is that for every $x \neq y$, $\Pr_{h,H}[h(x) = h(y) = H] = 1/K^2$ (or even just close to $1/K^2$).[11] Is this enough? Under this assumption, the probability that out of the satisfying assignments $\alpha_1,\ldots,\alpha_M$, only one

[11] This is slightly weaker than *pairwise independence*, a property which is commonplace in derandomization since it allows application of Chebyshev's inequality.

satisfies $h(\alpha_i) = H$, is at least

$$\sum_{i=1}^{M} \left(\Pr[h(\alpha_i) = H] - \sum_{\substack{j=1 \\ j \neq i}}^{M} \Pr[h(\alpha_i) = h(\alpha_j) = H] \right) = \frac{M}{K} - \frac{M(M-1)}{K^2} = \frac{M}{K}\left(1 - \frac{M-1}{K}\right)$$
$$\geq \frac{M}{K}\left(1 - \frac{M}{K}\right) = \frac{1}{4} - \left(\frac{1}{2} - \frac{M}{K}\right)^2.$$

The optimal value of $\theta = M/K$ is $1/2$, but this requires M to be a power of 2. Whenever $\theta < 1$, the probability above will be positive. Doubling K corresponds to halving θ. This always decreases the probability if $\theta \leq 1/2$. If $\theta > 1/2$, then this results in an improvement if $\theta - 1/2 > 1/2 - \theta/2$, that is, if $\theta > 2/3$. It follows that for every value of M, the optimal choice of K is the one which leads to $\theta \in [1/3, 2/3]$, in which case the probability above is at least $2/9$.

One problem with this approach is that we don't know the optimal value of K. However, there are only $n + O(1)$ possible values, and we can try each of them. As a result, the reduction only works with probability $\Omega(1/n)$. Is this good enough for reducing SAT to Unique-SAT?

One way to formalize this question is via Unique-SAT oracles. A Unique-SAT oracle gets as input a SAT instance, returns Yes if it has a unique satisfying assignment, returns No if it is unsatisfiable, and acts arbitrarily otherwise. In terms of this formalization, we can reduce SAT to Unique-SAT by performing the reduction above a (large) constant number of times for each value of K. If the original CNF is unsatisfiable, then the Unique-SAT oracle will always return No, while if the original CNF is satisfiable, the Unique-SAT oracle will return Yes on one of the runs with the correct value of K, with constant probability. We can increase the success probability by making more repeats.

In order to complete the description of the reduction, we need to specify a random ensemble of hash functions with the property that for every x, $\Pr_{h,H}[h(x) = H] = 1/K$, and for every $x \neq y$, $\Pr_{h,H}[h(x) = h(y) = H] = 1/K^2$, where $K = 2^k$. We choose a random linear hash function: $h(x) = Ax$, where A is a random $k \times n$ bit matrix, and arithmetic is performed modulo 2. The output $h(x)$ is a random vector in $\{0,1\}^k$, and accordingly, we choose H uniformly at random from $\{0,1\}^k$. The function h can clearly be computed efficiently, and so the constraint $h(x) = H$ can be encoded using polynomially many clauses (in fact, $O(kn)$ suffice).

Choosing H at random ensures that $\Pr[h(x) = H] = 2^{-k}$. If $x \neq y$ then in order for both $h(x) = H$ and $h(x) = h(y)$ to hold, we need $h(x) = H$ as well as $h(x \oplus y) = 0$. Each element in the vector $A(x \oplus y)$ is the inner product of a completely random vector (a row of A) with a non-zero vector $(x \oplus y)$, and so is uniformly distributed. Therefore $\Pr[h(x) = h(y)] = 2^{-k}$. Whatever the value of $h(x)$ is, the probability that it equals H is 2^{-k}, and so $\Pr[h(x) = h(y) = H] = 2^{-2k}$.

Reducing to $\oplus$SAT What if we want a many-one reduction? We can obtain such a reduction if we replace Unique-SAT with $\oplus$SAT, in which the goal is to check whether the given SAT instance has an odd number of satisfying assignments.

The main observation is that given a CNF ϕ_1 with M_1 satisfying assignments and a CNF ϕ_2 with M_2 satisfying assignments, we can construct a CNF ϕ with $M_1 M_2$ satisfying assignments by simply joining ϕ_1 and ϕ_2 together (taking their conjunction), using disjoint sets of variables.

This suggests that we find a reduction in which an unsatisfiable CNF maps to one with an *odd* number of assignments, while a satisfiable CNF maps (with some probability) to one with an *even* number of assignments. This is exactly the opposite parity than the Unique-SAT reduction: there, in the unsatisfiable case, the reduction constructed a CNF with zero satisfying assignments, while in the satisfiable case, the reduction constructed (with some probability) a CNF with a single satisfying assignment. We can fix this using a gadget which adds exactly one satisfying assignment to a given CNF ϕ on the variables $x_1, \ldots, x_n$:

$$(y \wedge \phi(x_1, \ldots, x_n)) \vee (y = x_1 = \cdots = x_n = 0).$$

Given a CNF ϕ, we first apply the Unique-SAT reduction to produce S many CNFs $\phi_1, \ldots, \phi_S$ with the following promise: if ϕ is unsatisfiable then so are $\phi_1, \ldots, \phi_S$, and if ϕ is satisfiable, then each ϕ_i has a unique satisfying assignment with probability $\Omega(1/n)$. Now we add a satisfying assignment to each of $\phi_1, \ldots, \phi_S$, obtaining $\phi'_1, \ldots, \phi'_S$. Finally, we put all of them together in a single CNF ψ, using dedicated variables for each ϕ'_i. If ϕ is unsatisfiable then ψ always has a unique satisfying assignment, whereas if ϕ is satisfiable then ψ has an even number of satisfying assignments with probability $1 - (1 - \Omega(1/n))^S = 1 - e^{-\Omega(S/n)}$, which can be made exponentially close to 1 by taking $S = n^2$.

We do not know how to construct a *deterministic* reduction of this sort from SAT to $\oplus$SAT.

7.2 Approximate counting and sampling

Similar techniques can be used for two natural problems: *approximate counting* and *approximate sampling*. Given a CNF ϕ, the approximate counting problem asks us to estimate the number of satisfying assignments of ϕ, say up to a small multiplicative error. The approximate sampling problem asks us to sample a satisfying assignment in an approximately uniform way, say each of the M satisfying assignments is sampled with probability $(1 \pm \epsilon)/M$ (this only makes sense if ϕ is satisfiable).

As stated, approximate counting is harder than satisfiability, so let us assume that we have a SAT oracle at our disposal. How do we use it to approximately count the number M of satisfying assignments of a given satisfiable CNF ϕ? Recall that for every $K = 2^k$, the reduction in the preceding section constructed a CNF ψ which has a unique satisfying assignment with probability at least $M/K(1 - M/K)$. In particular, if we denote by p_k the probability that ψ is satisfiable with this setting of k, then

$$\frac{M}{K}\left(1 - \frac{M}{K}\right) \leq p_k \leq \frac{M}{K},$$

the upper bound following from a union bound.

If $M = 2^m$ then

$$\frac{1}{4} \leq p_{m+1} \leq \frac{1}{2}, \quad \frac{3}{16} \leq p_{m+2} \leq \frac{1}{4}, \quad \frac{7}{64} \leq p_{m+3} \leq \frac{1}{8}.$$

Monotonicity shows that $p_k \geq 1/4$ for all $k \leq m$. Hence by estimating p_k well enough for all values of k up to $n + 3$, we can determine m (since we can tell apart $m + 2$ and $m + 3$).

In general, M need not be a power of 2, but similar calculations show that we can estimate it within some constant factor C (we skip the tedious details).

We can improve the approximation factor using the trick we used when reducing SAT to $\oplus$SAT: joining together S copies of ϕ on disjoint variables, we obtain a CNF with M^S satisfying assignments, and so applying the previous procedure, we obtain a $C^{1/S}$-approximation of M. For large S, the approximation ratio is $e^{\ln C/S} \approx 1 + (\ln C)/S$, and so we can get a $(1 + 1/n^D)$-approximation, for any constant D, using a polynomial number of calls to a satisfiability oracle.

Approximate counting with small error allows us to approximately sample. The idea is simple: given a CNF ϕ, by substituting $x_1 = 0$ and $x_1 = 1$ we can approximate the numbers of satisfying assignments M_0, M_1 with the given value of x_1. We choose x_1 to be b with probability $M_b/(M_0 + M_1)$, and continue to sample the remaining variables in the same way. If our estimates on M_0, M_1 are good enough, then the resulting sample will be close to uniform.

This is actually a general reduction from approximate sampling to approximate counting, due originally to Jerrum, Valiant and Vazirani [JVV86], improved by Bellare, Goldreich and Petrank [BGP00], who showed how to *exactly* sample a satisfying assignment.

There is also a reduction in the other direction The rough idea is that by sampling many satisfying assignments and observing the value of x_1, we can estimate the fraction of satisfying assignments with each value of x_1. We then fix x_1 to a value at random (with the same proportions), and continue in the same way. Multiplying the fractions encountered along the way gives an estimate for $1/M$.

References

[ABI+09] Eric Allender, Michael Bauland, Neil Immerman, Henning Schnoor, and Heribert Vollmer. The complexity of satisfiability problems: refining Schaefer's theorem. *J. Comput. System Sci.*, 75(4):245–254, 2009. doi:10.1016/j.jcss.2008.11.001.

[AFT11] Albert Atserias, Johannes Klaus Fichte, and Marc Thurley. Clause-learning algorithms with many restarts and bounded-width resolution. *J. Artificial Intelligence Res.*, 40:353–373, 2011. doi:10.1613/jair.3152.

[AL86] Ron Aharoni and Nathan Linial. Minimal non-two-colorable hypergraphs and minimal unsatisfiable formulas. *J. Combin. Theory Ser. A*, 43(2):196–204, 1986. doi:10.1016/0097-3165(86)90060-9.

[ALM+98] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and the hardness of approximation problems. *J. ACM*, 45(3):501–555, 1998. doi:10.1145/278298.278306.

[AM20] Albert Atserias and Moritz Müller. Automating resolution is NP-hard. *J. ACM*, 67(5):Art. 31, 17, 2020. doi:10.1145/3409472.

[AS98] Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs: a new characterization of NP. *J. ACM*, 45(1):70–122, 1998. doi:10.1145/273865.273901.

[AS00] Noga Alon and Benny Sudakov. Bipartite subgraphs and the smallest eigenvalue. *Combin. Probab. Comput.*, 9(1):1–12, 2000. doi:10.1017/S0963548399004071.

[BCH+96] Mihir Bellare, Don Coppersmith, Johan Håstad, Marcos Kiwi, and Madhu Sudan. Linearity testing in characteristic two. volume 42, pages 1781–1795. 1996. Codes and complexity. doi:10.1109/18.556674.

[BGP00] Mihir Bellare, Oded Goldreich, and Erez Petrank. Uniform generation of NP-witnesses using an NP-oracle. *Inform. and Comput.*, 163(2):510–526, 2000. doi:10.1006/inco.2000.2885.

[BKM21] Mark Braverman, Subhash Khot, and Dor Minzer. On rich 2-to-1 games. In *12th Innovations in Theoretical Computer Science Conference*, volume 185 of *LIPIcs. Leibniz Int. Proc. Inform.*, pages Art. No. 27, 20. Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2021.

[BKS04] Paul Beame, Henry Kautz, and Ashish Sabharwal. Towards understanding and harnessing the potential of clause learning. *J. Artificial Intelligence Res.*, 22:319–351, 2004. doi:10.1613/jair.1410.

[BLR93] Manuel Blum, Michael Luby, and Ronitt Rubinfeld. Self-testing/correcting with applications to numerical problems. *Journal of Computer and System Sciences*, 47(3):549–595, 1993. URL: https://www.sciencedirect.com/science/article/pii/002200009390044W, doi:https://doi.org/10.1016/0022-0000(93)90044-W.

[Bor75] Christer Borell. The Brunn-Minkowski inequality in Gauss space. *Invent. Math.*, 30(2):207–216, 1975. doi:10.1007/BF01425510.

[BS97] Roberto J. Bayardo and Robert C. Schrag. Using csp look-back techniques to solve real-world sat instances. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence and Ninth Conference on Innovative Applications of Artificial Intelligence*, AAAI'97/IAAI'97, page 203–208. AAAI Press, 1997.

[BS01] Eli Ben-Sasson. *Expansion in Proof Complexity*. PhD thesis, Hebrew University, 2001.

[BSIW04] Eli Ben-Sasson, Russell Impagliazzo, and Avi Wigderson. Near optimal separation of tree-like and general resolution. *Combinatorica*, 24(4):585–603, 2004. doi:10.1007/s00493-004-0036-5.

[BSW01] Eli Ben-Sasson and Avi Wigderson. Short proofs are narrow—resolution made simple. *J. ACM*, 48(2):149–169, 2001. doi:10.1145/375827.375835.

[Bul17] Andrei A. Bulatov. A dichotomy theorem for nonuniform CSPs. In *58th Annual IEEE Symposium on Foundations of Computer Science—FOCS 2017*, pages 319–330. IEEE Computer Soc., Los Alamitos, CA, 2017. doi:10.1109/FOCS.2017.37.

[CEI96] Matthew Clegg, Jeffery Edmonds, and Russell Impagliazzo. Using the Groebner basis algorithm to find proofs of unsatisfiability. In *Proceedings of the Twenty-eighth Annual ACM Symposium on the Theory of Computing (Philadelphia, PA, 1996)*, pages 174–183. ACM, New York, 1996. doi:10.1145/237814.237860.

[Che09] Hubie Chen. A rendezvous of logic, complexity, and algebra. *ACM Comput. Surv.*, 42(1), dec 2009. doi:10.1145/1592451.1592453.

[Dal00] Víctor Dalmau. *Computational Complexity of Problems over Generalized Formulas*. PhD thesis, Universitat Politécnica de Catalunya, 2000.

[DDG+17] Roee David, Irit Dinur, Elazar Goldenberg, Guy Kindler, and Igor Shinkar. Direct sum testing. *SIAM J. Comput.*, 46(4):1336–1369, 2017. doi:10.1137/16M1061655.

[Del73] P. Delsarte. An algebraic approach to the association schemes of coding theory. *Philips Res. Rep. Suppl.*, (10):vi+97, 1973.

[DGH+02] Evgeny Dantsin, Andreas Goerdt, Edward A. Hirsch, Ravi Kannan, Jon Kleinberg, Christos Papadimitriou, Prabhakar Raghavan, and Uwe Schöning. A deterministic $(2 - 2/(k + 1))^n$ algorithm for k-SAT based on local search. *Theoret. Comput. Sci.*, 289(1):69–83, 2002. doi:10.1016/S0304-3975(01)00174-8.

[Din07] Irit Dinur. The PCP theorem by gap amplification. *J. ACM*, 54(3):Art. 12, 44, 2007. doi:10.1145/1236457.1236459.

[DLL62] Martin Davis, George Logemann, and Donald Loveland. A machine program for theorem-proving. *Comm. ACM*, 5:394–397, 1962. doi:10.1145/368273.368557.

[DP60] Martin Davis and Hilary Putnam. A computing procedure for quantification theory. *J. Assoc. Comput. Mach.*, 7:201–215, 1960. doi:10.1145/321033.321034.

[DSS22] Jian Ding, Allan Sly, and Nike Sun. Proof of the satisfiability conjecture for large k. *Ann. of Math. (2)*, 2022+.

[FG01] Joel Friedman and Andreas Goerdt. Recognizing more unsatisfiable random 3-SAT instances efficiently. In *Automata, languages and programming*, volume 2076 of *Lecture Notes in Comput. Sci.*, pages 310–321. Springer, Berlin, 2001. URL: https://doi.org/10.1007/3-540-48224-5_26, doi:10.1007/3-540-48224-5_26.

[FGK05] Joel Friedman, Andreas Goerdt, and Michael Krivelevich. Recognizing more unsatisfiable random k-SAT instances efficiently. *SIAM J. Comput.*, 35(2):408–430, 2005. doi:10.1137/S009753970444096X.

[FK81] Z. Füredi and J. Komlós. The eigenvalues of random symmetric matrices. *Combinatorica*, 1(3):233–241, 1981. doi:10.1007/BF02579329.

[FO07] Uriel Feige and Eran Ofek. Easily refutable subformulas of large random 3CNF formulas. *Theory Comput.*, 3:25–43, 2007. doi:10.4086/toc.2007.v003a002.

[FS02] Uriel Feige and Gideon Schechtman. On the optimality of the random hyperplane rounding technique for MAX CUT. volume 20, pages 403–440. 2002. Probabilistic methods in combinatorial optimization. doi:10.1002/rsa.10036.

[FV99] Tomás Feder and Moshe Y. Vardi. The computational structure of monotone monadic SNP and constraint satisfaction: a study through Datalog and group theory. *SIAM J. Comput.*, 28(1):57–104, 1999. doi:10.1137/S0097539794266766.

[GK01] Andreas Goerdt and Michael Krivelevich. Efficient recognition of random unsatisfiable k-SAT instances by spectral methods. In *STACS 2001 (Dresden)*, volume 2010 of *Lecture Notes in Comput. Sci.*, pages 294–304. Springer, Berlin, 2001. URL: https://doi.org/10.1007/3-540-44693-1_26, doi:10.1007/3-540-44693-1_26.

[GL03] Andreas Goerdt and André Lanka. Recognizing more random unsatisfiable 3-SAT instances efficiently. In *Typical case complexity and phase transitions*, volume 16 of *Electron. Notes Discrete Math.*, page 26. Elsevier Sci. B. V., Amsterdam, 2003.

[GW95] Michel X. Goemans and David P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. Assoc. Comput. Mach.*, 42(6):1115–1145, 1995. doi:10.1145/227683.227684.

[Hae21] Willem H. Haemers. Hoffman's ratio bound. *Linear Algebra Appl.*, 617:215–219, 2021. doi:10.1016/j.laa.2021.02.010.

[Hås01] Johan Håstad. Some optimal inapproximability results. *J. ACM*, 48(4):798–859, 2001. doi:10.1145/502090.502098.

[Her14] Timon Hertli. 3-SAT faster and simpler—Unique-SAT bounds for PPSZ hold in general. *SIAM J. Comput.*, 43(2):718–729, 2014. doi:10.1137/120868177.

[IM99] Kazuo Iwama and Shuichi Miyazaki. Tree-like resolution is superpolynomially slower than DAG-like resolution for the pigeonhole principle. In *Algorithms and computation (Chennai, 1999)*, volume 1741 of *Lecture Notes in Comput. Sci.*, pages 133–142. Springer, Berlin, 1999. URL: https://doi.org/10.1007/3-540-46632-0_14, doi:10.1007/3-540-46632-0_14.

[JVV86] Mark R. Jerrum, Leslie G. Valiant, and Vijay V. Vazirani. Random generation of combinatorial structures from a uniform distribution. *Theoret. Comput. Sci.*, 43(2-3):169–188, 1986. doi:10.1016/0304-3975(86)90174-X.

[Kar72] Richard M. Karp. Reducibility among combinatorial problems. In *Complexity of computer computations (Proc. Sympos., IBM Thomas J. Watson Res. Center, Yorktown Heights, N.Y., 1972)*, pages 85–103, 1972.

[Kar99] Howard Karloff. How good is the Goemans-Williamson MAX CUT algorithm? *SIAM J. Comput.*, 29(1):336–350, 1999. doi:10.1137/S0097539797321481.

[Kho02] Subhash Khot. On the power of unique 2-prover 1-round games. In *Proceedings of the Thirty-Fourth Annual ACM Symposium on Theory of Computing*, pages 767–775. ACM, New York, 2002. doi:10.1145/509907.510017.

[KKMO07] Subhash Khot, Guy Kindler, Elchanan Mossel, and Ryan O'Donnell. Optimal inapproximability results for MAX-CUT and other 2-variable CSPs? *SIAM J. Comput.*, 37(1):319–357, 2007. doi:10.1137/S0097539705447372.

[KMS18] Subhash Khot, Dor Minzer, and Muli Safra. Pseudorandom sets in Grassmann graph have near-perfect expansion. In *59th Annual IEEE Symposium on Foundations of Computer Science—FOCS 2018*, pages 592–601. IEEE Computer Soc., Los Alamitos, CA, 2018. doi:10.1109/FOCS.2018.00062.

[Kro67] M. R. Krom. The decision problem for a class of first-order formulas in which all disjunctions are binary. *Z. Math. Logik Grundlagen Math.*, 13:15–20, 1967. doi:10.1002/malq.19670130104.

[Lad75] Richard E. Ladner. On the structure of polynomial time reducibility. *J. Assoc. Comput. Mach.*, 22:155–171, 1975. doi:10.1145/321864.321877.

[Lov79] László Lovász. On the Shannon capacity of a graph. *IEEE Trans. Inform. Theory*, 25(1):1–7, 1979. doi:10.1109/TIT.1979.1055985.

[LY20] Andrea Lincoln and Adam Yedidia. Faster random k-CNF satisfiability. In *47th International Colloquium on Automata, Languages, and Programming*, volume 168 of *LIPIcs. Leibniz Int. Proc. Inform.*, pages Art. No. 78, 12. Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2020.

[MMZ06] Stephan Mertens, Marc Mézard, and Riccardo Zecchina. Threshold values of random K-SAT from the cavity method. *Random Structures Algorithms*, 28(3):340–373, 2006. doi:10.1002/rsa.20090.

[MOO10] Elchanan Mossel, Ryan O'Donnell, and Krzysztof Oleszkiewicz. Noise stability of functions with low influences: invariance and optimality. *Ann. of Math. (2)*, 171(1):295–341, 2010. doi:10.4007/annals.2010.171.295.

[Mos10] Elchanan Mossel. Gaussian bounds for noise correlation of functions. *Geom. Funct. Anal.*, 19(6):1713–1756, 2010. doi:10.1007/s00039-010-0047-x.

[MPZ02] M. Mézard, G. Parisi, and R. Zecchina. Analytic and algorithmic solution of random satisfiability problems. *Science*, 297(5582):812–815, 2002. URL: https://www.science.org/doi/abs/10.1126/science.1073287, arXiv:https://www.science.org/doi/pdf/10.1126/science.1073287, doi:10.1126/science.1073287.

[MR99] Sanjeev Mahajan and H. Ramesh. Derandomizing approximation algorithms based on semidefinite programming. *SIAM J. Comput.*, 28(5):1641–1663, 1999. doi:10.1137/S0097539796309326.

[MS11] Robin A. Moser and Dominik Scheder. A full derandomization of Schöning's k-SAT algorithm. In *STOC'11—Proceedings of the 43rd ACM Symposium on Theory of Computing*, pages 245–251. ACM, New York, 2011. doi:10.1145/1993636.1993670.

[MSS99] João P. Marques-Silva and Karem A. Sakallah. GRASP: a search algorithm for propositional satisfiability. *IEEE Trans. Comput.*, 48(5):506–521, 1999. doi:10.1109/12.769433.

[MTz14] Sebastian Müller and Iddo Tzameret. Short propositional refutations for dense random 3CNF formulas. *Ann. Pure Appl. Logic*, 165(12):1864–1918, 2014. doi:10.1016/j.apal.2014.08.001.

[MVV87] Ketan Mulmuley, Umesh V. Vazirani, and Vijay V. Vazirani. Matching is as easy as matrix inversion. *Combinatorica*, 7(1):105–113, 1987. doi:10.1007/BF02579206.

[O'D14] Ryan O'Donnell. *Analysis of Boolean functions*. Cambridge University Press, New York, 2014. doi:10.1017/CBO9781139814782.

[OW08] Ryan O'Donnell and Yi Wu. An optimal SDP algorithm for max-cut, and equally optimal long code tests. In *STOC'08*, pages 335–344. ACM, New York, 2008. doi:10.1145/1374376.1374425.

[PD11] Knot Pipatsrisawat and Adnan Darwiche. On the power of clause-learning SAT solvers as resolution engines. *Artificial Intelligence*, 175(2):512–525, 2011. doi:10.1016/j.artint.2010.10.002.

[Pos41] Emil L. Post. *The Two-Valued Iterative Systems of Mathematical Logic*. Annals of Mathematics Studies, No. 5. Princeton University Press, Princeton, N. J., 1941.

[PPSZ05] Ramamohan Paturi, Pavel Pudlák, Michael E. Saks, and Francis Zane. An improved exponential-time algorithm for k-SAT. *J. ACM*, 52(3):337–364, 2005. doi:10.1145/1066100.1066101.

[PPZ99] Ramamohan Paturi, Pavel Pudlák, and Francis Zane. Satisfiability coding lemma. *Chicago J. Theoret. Comput. Sci.*, pages Article 11, 19, 1999.

[Rag08] Prasad Raghavendra. Optimal algorithms and inapproximability results for every CSP? [extended abstract]. In *STOC'08*, pages 245–254. ACM, New York, 2008. doi:10.1145/1374376.1374414.

[Rao11] Anup Rao. Parallel repetition in projection games and a concentration bound. *SIAM J. Comput.*, 40(6):1871–1891, 2011. doi:10.1137/080734042.

[Raz87] A. A. Razborov. Lower bounds on the dimension of schemes of bounded depth in a complete basis containing the logical addition function. *Mat. Zametki*, 41(4):598–607, 623, 1987.

[Raz98] Ran Raz. A parallel repetition theorem. *SIAM J. Comput.*, 27(3):763–803, 1998. doi:10.1137/S0097539795280895.

[Sch78] Thomas J. Schaefer. The complexity of satisfiability problems. In *Conference Record of the Tenth Annual ACM Symposium on Theory of Computing (San Diego, Calif., 1978)*, pages 216–226. ACM, New York, 1978.

[Sch79] Alexander Schrijver. A comparison of the Delsarte and Lovász bounds. *IEEE Trans. Inform. Theory*, 25(4):425–429, 1979. doi:10.1109/TIT.1979.1056072.

[Sch02] U. Schöning. A probabilistic algorithm for k-SAT based on limited local search and restart. *Algorithmica*, 32(4):615–623, 2002. doi:10.1007/s00453-001-0094-7.

[Sch21] Dominik Scheder. PPSZ is better than you think. In *62nd Annual IEEE Symposium on Foundations of Computer Science (FOCS 2021)*. 2021.

[Smo87] R. Smolensky. Algebraic methods in the theory of lower bounds for boolean circuit complexity. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, STOC '87, page 77–82, New York, NY, USA, 1987. Association for Computing Machinery. doi:10.1145/28395.28404.

[STs74] V. N. Sudakov and B. S. Tsirel′son. Extremal properties of half-spaces for spherically invariant measures. *Zap. Naučn. Sem. Leningrad. Otdel. Mat. Inst. Steklov. (LOMI)*, 41:14–24, 165, 1974. Problems in the theory of probability distributions, II.

[Ta-15] Noam Ta-Shma. A simple proof of the isolation lemma. *Electron. Colloquium Comput. Complex.*, page 80, 2015. URL: https://eccc.weizmann.ac.il/report/2015/080.

[Tsei68] G. S. Tseitin. The complexity of a deduction in the propositional predicate calculus. *Zap. Naučn. Sem. Leningrad. Otdel. Mat. Inst. Steklov. (LOMI)*, 8:234–259, 1968.

[VV86] L. G. Valiant and V. V. Vazirani. NP is as easy as detecting unique solutions. *Theoret. Comput. Sci.*, 47(1):85–93, 1986. doi:10.1016/0304-3975(86)90135-0.

[VW19] Nikhil Vyas and Ryan Williams. On super strong ETH. In *Theory and applications of satisfiability testing—SAT 2019*, volume 11628 of *Lecture Notes in Comput. Sci.*, pages 406–423. Springer, Cham, 2019. URL: https://doi.org/10.1007/978-3-030-24258-9_28, doi:10.1007/978-3-030-24258-9_28.

[Zhu20] Dmitriy Zhuk. A proof of the CSP dichotomy conjecture. *J. ACM*, 67(5):Art. 30, 78, 2020. doi:10.1145/3402029.