Advanced Proof-Systems — Winter 2024

Problem Set 2

Lecturer: Ron Rothblum — Due: 28/2/2024

1 Better All-Zero Check

In class we saw an interactive protocol for checking that $\forall x \in \{0,1\}^n$ it holds that $P(x) = 0$ where P is an individual degree d polynomial over a field $\mathbb{F}$ (this was used in our proof that **coNP** $\subseteq$ **IP**). This protocol was via a reduction to *sumcheck*. The reduction crucially relied on the fact that $\mathbb{F}$ is a prime order field of size[1] greater than 2^n.

In this exercise you will show an alternate reduction that can use a much smaller (and non-prime) field.

1.1 Polynomial Identity Lemma

Recall that the fundamental theorem of algebra states that a non-zero univariate polynomial of degree d, cannot have more than d zeros. (You probably saw this for complex/real numbers but its also true for finite fields.) The following lemma[2] We start with a basic fact that bounds the number of roots of *multivariate* polynomials (extending the fundamental theorem of algebra which bounds the roots of a *univariate* polynomial).

The *total degree* of a monomial of the form $x^i y^j z^k$ is defined as $i+j+k$ (and similarly for more variables). The total degree of a polynomial is the maximal total degree of all of its monomials. For example, the polynomial $x^2y^3 + x^7z + 8$ has total degree 8.[3]

Lemma 1.1 (Polynomial Identity Lemma)**.** *Let $P : \mathbb{F}^n \to \mathbb{F}$ be an n-variate total degree d polynomial over a finite field $\mathbb{F}$. If P is not the all-zeros polynomial then:*

$$\Pr_{z \in \mathbb{F}^n}[P(z) = 0] \leq \frac{d}{|\mathbb{F}|}.$$

1. Prove the Polynomial Identity Lemma.
   Hint: By induction on n.

2. Show that the lemma is tight by exhibiting, for every $n \in \mathbb{N}$ and $d \leq |\mathbb{F}| - 1$, a total degree d polynomial for which the inequality in Lemma 1.1 is an equality.

[1] More specifically, we needed a field of *characteristic* greater than 2^n.

[2] The lemma is often referred to in the literature as the *Schwartz–Zippel lemma*, but the attribution to Schwartz and Zippel is debatable.

[3] In contrast, the *individual* degree of a polynomial is that largest degree of any variable that appears in it. Note that an n-variate polynomial with individual degree d has total degree at most $n \cdot d$.

1.2 All-Zero Check with Small Field

Lemma 1.2. *For every*[4] *finite field $\mathbb{F}$, there exists an interactive protocol between a prover P, which gets as input a polynomial $Q : \mathbb{F}^n \to \mathbb{F}$ of individual degree d, and a verifier V that gets as input oracle access to Q such that:*

- **Completeness:** *if for every $x \in \{0,1\}^n$ it holds that $Q(x) = 0$ then V accepts with probability 1 when interacting with P.*
- **Soundness:** *if there exists $x \in \{0,1\}^n$ such that $Q(x) \neq 0$ then for every P^*, the probability that V accepts when interacting with P^* is at most $O\left(\frac{d \cdot n}{|\mathbb{F}|}\right)$.*
- **Complexity:** *the verifier's running time (and the communication complexity) are at most $\mathrm{poly}(n, d, \log(|\mathbb{F}|))$.*

Prove Lemma 1.2. How many queries does the verifier make to Q?

Guideline 1.3.

1. *Construct an individual degree 1 (aka multi-linear) polynomial $I : \mathbb{F}^{2n} \to \mathbb{F}$ such that for every $x, x' \in \{0,1\}^n$:*
$$I(x, x') = \begin{cases} 1 & x = x' \\ 0 & x \neq x' \end{cases}$$
Note that for non-Boolean values (i.e., $x, x' \in \mathbb{F}^n \setminus \{0,1\}^n$) we make no explicit requirement on the value of $I(x, x')$ (except those arising from the fact that I is multi-linear).

   Hint: *for $n = 1$ take $I_1(x, x') = 1 - x - x' + 2 \cdot x \cdot x'$) (where 2 refers not to the integer 2 but rather the field element $1 + 1$).*[5] *For larger values of n take $I(x, x') = \prod_{i \in [n]} I_1(x_i, x'_i)$.*

2. *Observe that if $Q(x)$ vanishes (i.e., is 0) on all of $\{0,1\}^n$ then for every value of $z \in \mathbb{F}^n$ it holds that $\sum_{x \in \{0,1\}^n} Q_z(x) = 0$, where $Q_z(x) = Q(x) \cdot I(x, z)$.*

3. *Using the polynomial identity lemma, show that if $Q(x)$ does not vanish on all of $\{0,1\}^n$, then with high probability over $z \in \mathbb{F}^n$, it holds that $\sum_{x \in \{0,1\}^n} Q_z(x) \neq 0$.*

4. *Use the sumcheck protocol!*

2 Hardness of Approximating Clique Size

Deciding whether a given graph has a clique of size k is known to be NP-hard. The goal of this exercise is to show that this problem is also to hard to even approximate.

Given a parameter $\varepsilon \in [0, 1]$, we define a promise problem $\mathsf{GapClique}_\varepsilon$ as follows. The input to the problem is a graph G and an integer k. YES instances are graphs that contain a clique of size k. NO instances are graphs for which every clique is of size at most εk. An algorithm for the problem is one that accepts every YES instance and rejects every NO instance.

Use the PCP Theorem to show that $\mathsf{GapClique}_{1/2}$ is NP hard.

[4] As usual, assuming field operations can be computed in polylog($|\mathbb{F}|$) time.

[5] For those wondering how this mysterious polynomial was found, the answer is interpolation.

Guideline 2.1.

1. *Start with an arbitrary* NP *language L. The goal is to show a (Karp) reduction from L to* $\mathsf{GapClique}_{1/2}$.

2. *Given an input x for L, consider the* PCP *verifier V_x for L that makes q queries and uses r random bits (and has x already hardcoded).*

3. *Construct a graph on a subset of $\{0,1\}^q \times \{0,1\}^r$ (i.e., at most 2^{q+r} vertices) that includes only vertices (ω, ρ) such that V_x accepts the answers $\omega \in \{0,1\}^q$ when using the random string $\rho \in \{0,1\}^q$.*

4. *Put an edge between two vertices iff they are consistent: namely, the answers of neighboring vertices should agree on each common query.*

5. *Show that there exists an integer k such that if $x \in L$ then the graph has a clique of size k whereas if $x \notin L$ then it can have a clique of size at most $k/2$. Explain why this yields a Karp reduction from L to* GapClique *and conclude that the problem is* **NP**-*hard.*

Remark 2.2. *A better inapproximability result can be obtained by using a* PCP *with smaller soundness error and even better results (optimal) can be obtained via more advanced techniques.*

Remark 2.3. *This "exercise" won the 2001 Gödel prize (which is the most prestigious prize in theoretical computer science).*

3 Random Linear Codes (aka Gilbert–Varshamov Bound)

The goal of this question is to show that if we choose a code at random, from the set of all possible linear codes, then with high probability we'll get a code that has good relative distance.

Let $A \in_R \{0,1\}^{n \times k}$ and consider the code $C : \{0,1\}^k \to \{0,1\}^n$ defined as $C(x) = Ax$. Let $\varepsilon > 0$ and assume that $n = \Omega(k/\varepsilon^2)$. Show that with probability 0.99, the code C has relative distance at least $1/2 - \varepsilon$.

Guideline 3.1. *Use the probabilistic method. That is, show that the probability that any individual message is mapped to a low weight codeword is very small and then apply the union bound.*

Remark 3.2. *Observe that using a random linear code, we have obtained a binary code (that is, over alphabet $\{0,1\}$) that has relative distance which almost matches that of the Hadamard code (which has relative distance exactly $1/2$), but with* constant rate *(whereas the rate of the Hadamard code is exponentially vanishing). In other words, with exponentially less redundancy, we have achieved almost optimal distance. Codes that have constant rate and constant distance are often called "good" codes.*

Unfortunately, while this exercise establishes the existence *of good codes (over the binary alphabet), it does not indicate to us that any one particular code that we can point out is in fact good (even though almost all linear codes are good!).*[6] *Nevertheless, we mention that explicit good codes are known (but are somewhat more complicated) although explicit codes that* exactly *match the rate/distance tradeoff of random linear codes are not known.*

[6]Hence, a good analogy is this task is like "finding hay in a haystack".