Advanced Proof-Systems	Winter 2020-2021
Problem Set 3	
Lecturer: Ron Rothblum	Due: 30/12/2023

1 MIP vs PCP

The goal of this question is to exhibit a close relationship between PCPs and a natural extension of interactive proofs in which the verifier interacts with multiple non-communicating provers.

Definition 1.1. *A* multi-prover interactive proof *(*MIP*) for a language L consists of a probabilistic polynomial-time verifier V and k (computationally unbounded) provers $P_1, \dots, P_k$. Given a common input x, the verifier V is allowed to interact with each of the provers P_i. The provers are* not *allowed to interact with each other. We require that:*

- **Completeness:** *if $x \in L$ then V accepts with probability 1 when interacting with $P_1, \dots, P_k$.*
- **Soundness:** *if $x \notin L$, then for every $P_1^*, \dots, P_k^*$ the verifier accepts with probability at most $1/2$ when interacting with $P_1^*, \dots, P_k^*$.*

We say that an MIP is 2-message if it consists of a first message in which V sends queries to each of the k provers (simultaneously) and then a second message in which V gets back answers from each prover and then decides whether to accept or reject. We denote the length of messages sent from the verifier to each of the provers by ℓ_V and the length of messages sent back from the provers by ℓ_P.

Prove the following two claims:

1. If L has a k-prover 2-message MIP in which the verifier messages have length ℓ_V and the prover messages have length ℓ_P, then L has a $(k \cdot \ell_P)$-query PCP where the PCP proof string has length $k \cdot 2^{\ell_V} \cdot \ell_P$.

2. If L has a q-query (non-adaptive) PCP in which the PCP has length at most m, then L has a poly(q)-prover 2-message MIP in which the verifier messages have length $O(\log m)$ and prover messages have length 1 (i.e., a single bit).

   Simplifying assumption: you may assume for simplicity that each of the PCP verifier's queries *by itself* is uniformly distributed (but the q queries may still be highly correlated).

   Hint: have the provers answer according to the PCP but check consistency.

Guideline 1.2 (For Part 2)**.**

1. *First construct an* MIP *with q provers but very large soundness error $1 - \frac{1}{\text{poly}(q)}$ (later we will reduce the soundness error by repetition).*

2. *The q honest* MIP *provers answer according to the* PCP*.*

3. *With probability $1/2$, the* MIP *verifier simply runs the* PCP *verifier.*

4. *With probability* $1/2$*, the* MIP *verifier chooses a random coordinate* $i \in [m]$*, where* m *is the length of the* PCP *string, and sends* i *to all of the provers. It accepts iff all the provers answer with the same value.*

5. *To analyze soundness, fix* $x \notin L$ *and (wlog deterministic) cheating provers* $P_1^*, \ldots, P_k^*$*. Analyze the following two cases separately:*

   (a) *There exists a pair of provers that answer more than a* $\frac{1}{100q^2}$ *fraction of their queries* inconsistently *with one another.*

   (b) *There does not exist such a pair of provers. To argue soundness in this case, first argue that on most coordinates* all *of the provers agree. Then, use this fact to construct an accepting* PCP *for* x*.*

6. *Finally, repeat the entire proof systems* $\mathrm{poly}(q)$ *times (with* $\mathrm{poly}(q)$ *sets of* q *provers) to obtain constant soundness error.*

Remark 1.3. *One can get rid of the simplifying assumption by choosing* i *in the consistency check differently - generate the* q PCP *queries by running the verifier and then output one of these* q *queries at random.*

2 Tensor Codes

This exercise introduces and discusses the *tensoring* operation of codes. In a nutshell, this is a way to take two (linear) codes and combine them in an interesting way to derive a new code, inheriting some of the properties of the base codes and also magically generating some new properties.

Let $\mathbb{F}$ be a finite field, and let $C_1 : \mathbb{F}^{k_1} \to \mathbb{F}^{n_1}$ and $C_2 : \mathbb{F}^{k_2} \to \mathbb{F}^{n_2}$ be linear codes. We construct a new code $C^{\otimes} : \mathbb{F}^{k_1 \times k_2} \to \mathbb{F}^{n_1 \times n_2}$ as follows: view the input as a $k_1 \times k_2$ sized matrix. Encode each one of the k_1 rows using C_2 to get a matrix of size $k_1 \times n_2$. Now encode each one of these n_2 columns to get the codeword (a matrix of size $n_1 \times n_2$). The resulting code is denoted by $C_1 \otimes C_2$.

Notation. Consider a codeword $c \in C_1 \otimes C_2$. Since we view c as a matrix, it will be convenient to index its entries accordingly. Thus, we use $c(i_1, i_2)$ to denote the (i_1, i_2) entry of this matrix.

Also, following the coding literature, if C is a code, we sometimes abuse notation and use C to refer both to the function (mapping messages to codewords) and to the corresponding set of codewords (i.e., the image of the function).

2.1 A Characterization

1. Consider the set $\mathcal{M}$ of matrices $\mathbb{F}^{n_1 \times n_2}$ such that all of the rows are codewords of C_2 and all of the columns are codewords of C_1. Show that $C_1 \otimes C_2 \subseteq \mathcal{M}$.

   Hint: It may be useful to first show that the exact same code would be obtained if you first encoded the columns and then the rows.

2. Show that $\mathcal{M} \subseteq C_1 \otimes C_2$.

   Hint: Recall that for linear codes, any linear combination of codewords is also a codeword.

3. Deduce that $\mathcal{M} = C_1 \otimes C_2$.

   Hint: Don't forget your Matka!

2.2 Rate

Let $r_1 = k_1/n_1$ and $r_2 = k_2/n_2$ denote the rates of C_1 and C_2, respectively. Show that $C^{\otimes}$ is (1) a linear code and (2) has rate $r_1 \cdot r_2$.

2.3 Distance

Let $\delta_1, \delta_2 \in (0,1)$ be the minimal relative distances of C_1 and C_2, respectively. Show that $C^{\otimes}$ has relative distance at least $\delta_1 \cdot \delta_2$.

Hint: Recall that for linear codes it suffices to bound the minimal number of zeros in a non-zero codeword.

In the following we use the notation $C^{\otimes d} = C \otimes C^{\otimes d-1}$, where $C^{\otimes 1} = C$. The code $C^{\otimes d}$ is sometimes called the d-th dimensional tensor of C. Note that if C has message length k and codeword length n then $C^{\otimes d}$ has message length k^d and codeword length n^d. Indeed, it is often more convenient to view the messages and codewords of $C^{\otimes d}$ as "d-dimensional" matrices aka tensors (rather than viewing them as vectors), .[1] In terms of notation, similarly to the case of $d = 2$ discussed above, for $c \in C^{\otimes d}$ and $i_1, \ldots, i_d \in [n]$ we use $c(i_1, \ldots, i_d)$ to denote the corresponding entry of c.

2.4 Tensor of Reed-Solomon: NOT TO BE SUBMITTED

Recall that the Reed-Solomon code $RS : \mathbb{F}^k \to \mathbb{F}^n$ consists of all degree $k-1$ univariate polynomials over $\mathbb{F}$. Describe in 10 words or less the image of the code $RS^{\otimes 2}$ (without referring to the tensoring operation of course).

2.5 Sumcheck for Tensor Codes

Let $C_{\mathsf{base}} : \mathbb{F}^k \to \mathbb{F}^n$ be a code and consider the code $C : \mathbb{F}^{k^d} \to \mathbb{F}^{n^d}$ defined as $C = (C_{\mathsf{base}})^{\otimes d}$. In this question we will show that C supports a "sumcheck-like" protocol - namely, a protocol for verifying the sum of some product-set of the codeword elements.

Let $S \subseteq [n]$. Construct an interactive proof for verifying the sum over the elements of $c \in C$ belonging to the set S^d: that is, an interactive proof for the statement $\sum_{i_1,\ldots,i_d \in [S]} c(i_1, \ldots, i_d) = \alpha$, for a given $\alpha \in \mathbb{F}$. That is, show an interactive proof in which the verifier has oracle access to $c \in C$ and explicitly given α and the prover is given both c and α. The verifier should accept if the claim is true and should reject with probability at least δ^d if the claim is false, no matter what the prover does (in all cases you may assume that c is indeed a valid codeword). The verifier's work should be $\mathrm{poly}(d, n, \log(|\mathbb{F}|))$.

[1] For $d > 3$, best not to try to visualize them ☺.

Hint: It may be useful to first think of the case $d = 2$, in which case a useful proof string to consider is $w \in \mathbb{F}^n$ defined as $w_\lambda = \sum_{i_2 \in S} c(\lambda, i_2)$, for every $\lambda \in [n]$. Handle larger dimensions by recursion.

Remark 2.1. *The detection probability δ^d may seem very low, but is actually fine as long as we choose the base code to have distance $\delta \gg 1 - 1/d$.*

Remark 2.2. *Note the remarkable fact that even if we start with a base code that does not support a "sumcheck-like" operation, the resulting tensor code magically does!*

Similarly, even if we start with a base code that is not locally-testable, the resulting tensor code[2] *turns out to be locally testable.*

2.6 Reflection: NOT TO BE SUBMITTED

Try to see how the sumcheck protocol that we saw for polynomials is a special case of the protocol for general tensor codes.

[2] Actually, this fails for $d = 2$, but works for $d \geq 3$.