1 Conclusions

Some main points to take away from this course:

- There is strong evidence that randomized algorithms are not significantly more powerful than deterministic algorithms. However, we currently only know how to prove this in general (e.g. \( \text{BPP} = \text{P} \)) based on other conjectures in complexity theory (the existence of sufficiently hard functions in \( \text{E} \)).

- The pseudorandom objects we studied have many other applications in theoretical computer science beyond simply eliminating randomness.

- There are deep connections between the pseudorandom objects, as reviewed more formally below.

We now present all of our main objects of study (expanders, extractors, samplers, list-decodable codes, and black-box PRG constructions) in the ‘list-decoding’ framework we used in Lecture Notes 16. All of these objects can be presented as functions \( \Gamma : [N] \times [D] \to [D] \times [M] \). (In some cases, the output is more naturally viewed as a single element rather than a pair.) For a set \( T \subseteq [D] \times [M] \) and \( \varepsilon \geq 0 \), we define \( \text{LIST}(T; \varepsilon) = \{ x \in [N] : \Pr_y[\Gamma(x, y)] \geq \varepsilon \} \).

Then all of our objects can be presented as follows.

Expanders.

- \( \Gamma(x, y) \) is the \( y \)'th neighbor of \( x \).
- Restrict to \( T \) of size less than \( KA \), where \( A \) is the expansion factor.
- Require that for every such \( T \), \( \vert \text{LIST}(T, 1) \vert < K \).

Extractors.

- \( \Gamma(x, y) = \text{Ext}(x, y) \).
- Consider all sets \( T \).
- Require that for every \( T \), \( \vert \text{LIST}(T; \mu(T) + \varepsilon) \vert < K \), where \( k = \log K \) is (roughly) the min-entropy threshold for the extractor.
Black-Box PRG Constructions.

- \( \Gamma(x, y) = G^x(y) \) is the output of the PRG when \( x \) is the truth-table of the hard function and \( y \) is the seed.
- Consider all sets \( T \).
- Require that each element of list \( \text{LIST}(T, \mu(T) + \varepsilon) \) can be efficiently locally decoded using an oracle to \( T \) and \( k = \log K \) bits of advice.

List-Decodable Codes.

- \( \Gamma(x, y) = (y, \text{Enc}(x)y) \).
- Restrict to \( T \) of the form \( T_r = \{(y, r(y)) : y \in [D]\} \) for a received word \( r : [D] \to [M] \).
- Require that for every \( r \), we have \( |\text{LIST}(T_r, 1/M + \varepsilon)| \leq K \). Here \( K \) is the bound on list size.
- Typically we want decoding to be efficient, in the sense that given \( r \), all of the elements of \( \text{LIST}(T_r, 1/M + \varepsilon) \) can be enumerated in polynomial time.

Black-Box Worst-Case to Average-Case Constructions.

- \( \Gamma(x, y) \) is \( \hat{f}(y) \), where \( \hat{f} \) is the average-case-hard function constructed from the worst-case hard function \( f_x \) whose truth table is \( x \).
- Restrict to \( T \) of the form \( T_r = \{(y, r(y)) : y \in [D]\} \) for a received word \( r : [D] \to [M] \).
- Require that for every \( r \), each element of list \( \text{LIST}(T_r, 1/M + \varepsilon) \) can be efficiently locally decoded using an oracle to \( r \) and \( k = \log K \) bits of advice.

In the rest of these notes, we survey some of the topics that we did not cover.

2 And Beyond

Some major topics we did not cover (to be surveyed in class):

- Are circuit lower bounds necessary for derandomization?
- Extractors and PRGs from Reed–Muller codes.
- Cryptographic pseudorandomness.
- Algebraic pseudorandomness.
- Hardness amplification.
- Derandomizing space-bounded computation.
- Deterministic extractors.