CS 579: Computational Complexity Fall 2025

Lecture 07 - September 16

Prof. Fernando Granha Jeronimo *Super Scribes: Connor Mowry & Abhinav Angirekula*

1 Recap: Previous Week's Topics

Last week while Fernando was away, we learned about two foundational topics that connect to our study of interactive proofs and PCPs: expander graphs and list decoding.

1.1 Expander Graphs

Expander graphs are sparse graphs with strong connectivity properties. They serve as fundamental pseudorandom objects in theoretical computer science with applications spanning:

- Derandomization and pseudorandom generators
- Error-correcting codes and their explicit constructions
- Network design and communication complexity
- The PCP theorem itself, where expanders enable gap amplification

1.2 List Decoding

List decoding provides a powerful generalization of unique decoding by allowing the decoder to output a small list of possible codewords rather than a single codeword.

Definition 1 (List Decoding)**.** A code C is (p, L)-list decodable if for every received word w, there are at most L codewords within relative distance p from w.

Key properties:

- **Unique decoding:** Special case where $L = 1$ and $p < d/2$ (where d is minimum distance)
- **List decoding advantage:** Can correct errors beyond the unique decoding radius $d/2$
- **List size:** Remarkably, L can remain polynomial (often constant) even for error rates approaching the information-theoretic limit
- **Hierarchy:** Unique Decoding $\subseteq$ List Decoding $\subseteq$ List Recovery

These tools become crucial in PCP constructions, where locally decodable codes with list-decoding properties enable efficient proof verification.

2 Introduction

In this lecture, we explore interactive proof systems and probabilistically checkable proofs (PCPs), two fundamental concepts that revolutionized our understanding of proof verification and approximation algorithms. We begin with the Graph Isomorphism problem as a motivating example, then formalize interactive proofs, and conclude with the powerful PCP theorem and its connections to hardness of approximation.

3 Graph Isomorphism

Definition 2 (Graph Isomorphism)**.** Two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are *isomorphic*, denoted $G_1 \cong G_2$, if there exists a bijection $\pi : V_1 \to V_2$ such that $(u, v) \in E_1$ if and only if $(\pi(u), \pi(v)) \in E_2$.

The Graph Isomorphism problem asks: given two graphs G_1 and G_2, determine whether $G_1 \cong G_2$.

Remark. Graph Isomorphism belongs to **NP**, since a permutation π serves as a polynomial-size certificate that can be verified in polynomial time. The exact complexity of this problem remains unknown—it is neither known to be in **P** nor proven to be **NP**-complete. This problem is not believed to be **NP**-complete, and Babai found a breakthrough quasi-polynomial-time algorithm for it.

3.1 Graph Non-Isomorphism

While Graph Isomorphism is in **NP**, its complement problem presents an interesting challenge:

Definition 3 (Graph Non-Isomorphism)**.** The Graph Non-Isomorphism problem asks: given graphs G_1 and G_2, verify that for all permutations π, the graphs are not the same up to relabeling.

Graph Non-Isomorphism appears to require checking all $n!$ possible permutations, suggesting it may not be in **NP**. However, as we'll see, it admits an elegant interactive proof.

4 The Polynomial Hierarchy

Before discussing interactive proofs, we briefly introduce the polynomial hierarchy, which provides context for understanding the power of interaction.

Definition 4 (Polynomial Hierarchy)**.** The *Polynomial Hierarchy* (PH) consists of a sequence of complexity classes defined using bounded alternation of quantifiers:

- $\Sigma_1^p = \mathbf{NP}$: Languages decidable with one existential quantifier
- $\Pi_1^p = \mathbf{coNP}$: Languages decidable with one universal quantifier
- For $k \geq 2$:

$$\Sigma_k^p = \{L : x \in L \iff \exists y_1 \forall y_2 \exists y_3 \cdots Q_k y_k : \varphi(x, y_1, \ldots, y_k)\} \tag{1}$$
$$\Pi_k^p = \{L : x \in L \iff \forall y_1 \exists y_2 \forall y_3 \cdots Q_k y_k : \varphi(x, y_1, \ldots, y_k)\} \tag{2}$$

  where $|y_i| = \text{poly}(|x|)$ and φ is computable in polynomial time.

The entire hierarchy is $\mathbf{PH} = \bigcup_{k \geq 1} \Sigma_k^p = \bigcup_{k \geq 1} \Pi_k^p$.

Example (Graph Non-Isomorphism in the Hierarchy)**.** Graph Non-Isomorphism can be expressed as:

$$\forall \pi \in S_n : \pi(G_1) \neq G_2$$

This places it in $\Pi_1^p = \mathbf{coNP}$, though as we'll see, interaction provides a more efficient verification.

Remark. Key observations about the polynomial hierarchy:

- Adjacent quantifiers of the same type can be merged: $\exists x \exists y$ is equivalent to $\exists (x, y)$

- Each level contains the previous: $\Sigma_k^p \cup \Pi_k^p \subseteq \Sigma_{k+1}^p \cap \Pi_{k+1}^p$
- **PH** $\subseteq$ **PSPACE**, though equality is unknown (and it is not expected)
- If **PH** collapses (i.e., $\Sigma_k^p = \Sigma_{k+1}^p$ for some k), then **PH** $= \Sigma_k^p$

Definition 5 (TQBF)**.** The *True Quantified Boolean Formula* (TQBF) problem asks whether a fully quantified Boolean formula is true:

$$\exists x_1 \forall x_2 \exists x_3 \cdots Q_n x_n : \varphi(x_1, \ldots, x_n)$$

where each $x_i \in \{0, 1\}$ and φ is a Boolean formula. TQBF is **PSPACE**-complete.

The polynomial hierarchy captures problems with bounded quantifier alternation, while interactive proofs, as we'll see next, achieve the full power of **PSPACE** through interaction and randomness.

5 Interactive Proof Systems

5.1 The Model

Definition 6 (Interactive Proof System)**.** An *interactive proof system* for a language L consists of two parties:

- A *Verifier*: A probabilistic polynomial-time algorithm with access to random bits
- A *Prover*: A computationally unbounded entity

After polynomial rounds of interaction, the Verifier accepts or rejects based on the input and the conversation transcript.

Definition 7 (The Class **IP**)**.** A language $L \in$ **IP** if there exists an interactive proof system such that:

- **Completeness:** If $x \in L$, there exists a Prover strategy that causes the Verifier to accept with probability $\geq 2/3$
- **Soundness:** If $x \notin L$, for all Prover strategies, the Verifier rejects with probability $\geq 2/3$

5.2 Protocol for Graph Non-Isomorphism

We now present an interactive protocol demonstrating that Graph Non-Isomorphism $\in$ **IP**.

Theorem 8. *Graph Non-Isomorphism* $\in$ **IP**.

Proof. We describe a protocol between Verifier and Prover:

Protocol:

1. Verifier samples $i \in \{1, 2\}$ uniformly at random
2. Verifier samples a random permutation π and computes $H = \pi(G_i)$
3. Verifier sends H to the Prover
4. Prover responds with $j \in \{1, 2\}$
5. Verifier accepts if and only if $j = i$

Analysis:

Case 1: $G_1 \not\cong G_2$ (graphs are non-isomorphic)

- The all-powerful Prover can determine which graph was permuted
- For any H, either $H \cong G_1$ or $H \cong G_2$, but not both
- The Prover always correctly identifies i
- Pr[Verifier accepts] $= 1$

Case 2: $G_1 \cong G_2$ (graphs are isomorphic)

- Any permutation of G_1 is also a permutation of G_2
- The distribution of H is identical regardless of whether $i = 1$ or $i = 2$
- No Prover strategy can determine i with probability $> 1/2$
- Pr[Verifier accepts] $\leq 1/2$

□

5.3 Power of Interaction

The Graph Non-Isomorphism protocol illustrates how interaction with randomness enables verification of statements that seem to require exponential certificates. This leads to a remarkable characterization:

Theorem 9 (Shamir, 1992)**.** $\mathbf{IP} = \mathbf{PSPACE}$.

This theorem shows that interactive proofs with polynomial rounds are exactly as powerful as polynomial space computation.

6 Probabilistically Checkable Proofs

6.1 Definition and Parameters

While interactive proofs allow multiple rounds of communication, PCPs take a different approach: the proof is written down once, but the verifier can query it probabilistically.

Definition 10 (PCP)**.** A language $L \in \mathbf{PCP}[r(n), q(n)]$ if there exists a probabilistic polynomial-time verifier V such that:

- On input x of length n, V uses $O(r(n))$ random bits
- V makes $O(q(n))$ queries to a proof string π
- **Completeness:** If $x \in L$, $\exists \pi$ such that $\Pr[V^\pi(x) = 1] = 1$
- **Soundness:** If $x \notin L$, $\forall \pi$, $\Pr[V^\pi(x) = 1] \leq 1/2$

Remark. Note that $\mathbf{NP} = \mathbf{PCP}[0, \text{poly}(n)]$, as traditional NP verification uses no randomness and reads the entire proof.

6.2 The PCP Theorem

Theorem 11 (PCP Theorem (Arora, Safra 1992; Arora, Lund, Motwani, Sudan, Szegedy 1992))**.**

$$\mathbf{NP} = \mathbf{PCP}[O(\log n), O(1)]$$

This remarkable result states that every NP statement has a proof that can be verified probabilistically by:

- Using only logarithmically many random bits
- Reading only a constant number of proof bits
- Maintaining perfect completeness and constant soundness

7 Connection to Constraint Satisfaction Problems

7.1 CSP Formulation

Definition 12 (Constraint Satisfaction Problem)**.** A *Constraint Satisfaction Problem* (CSP) instance ϕ consists of:

- Variables $x_1, \ldots, x_n$ over domain $[q]$
- Constraints $C_1, \ldots, C_m$ where each C_i involves k variables
- For each constraint, a set of satisfying assignments

The *value* of ϕ, denoted $\text{val}(\phi)$, is the maximum fraction of simultaneously satisfiable constraints.

Example (MAX-3SAT)**.** Variables are Boolean, constraints are clauses with 3 literals. The goal is to maximize the fraction of satisfied clauses.

Example (MAX-CUT)**.** Given graph $G = (V, E)$, assign each vertex to one of two sets. Constraints correspond to edges; an edge is satisfied if its endpoints are in different sets.

7.2 PCP Implies Hardness of Approximation

The PCP Theorem has a remarkable equivalent formulation:

Theorem 13 (Gap-CSP Version of PCP Theorem)**.** *There exists a polynomial-time reduction from any NP language L to CSP instances such that:*

- *If $x \in L$: The produced CSP ϕ has* $\text{val}(\phi) = 1$
- *If $x \notin L$: The produced CSP ϕ has* $\text{val}(\phi) \leq 1/2$

Corollary 14. *Unless* $\mathbf{P} = \mathbf{NP}$*, there is no polynomial-time algorithm that approximates MAX-CSPs within a factor better than some constant $\alpha < 1$.*

7.3 From PCP Verifier to CSP

Given a PCP verifier V using r random bits and making q queries:

1. Create variables for each proof bit position

2. For each possible random string $R \in \{0,1\}^r$:
   - V with randomness R queries positions $i_1, \ldots, i_q$
   - Create a constraint on variables $x_{i_1}, \ldots, x_{i_q}$
   - The constraint is satisfied iff V accepts given those bit values

This construction yields:

- Number of constraints: $2^r = 2^{O(\log n)} = \text{poly}(n)$
- Variables per constraint: $q = O(1)$
- Completeness: If $x \in L$, the correct proof satisfies all constraints
- Soundness: If $x \notin L$, at most half the constraints can be satisfied

8 Implications and Open Problems

The PCP Theorem fundamentally changed our understanding of:

- **Proof verification:** Proofs can be verified locally with constant queries
- **Approximation algorithms:** Many optimization problems have inherent approximation barriers
- **Error-correcting codes:** PCPs connect to locally testable and decodable codes

8.1 Current Research Directions

1. **Unique Games Conjecture:** Predicts optimal approximation ratios for many problems
2. **Short PCPs:** Minimizing proof length while maintaining constant queries
3. **Quantum PCPs:** Extension to quantum proofs and verification
4. **Fine-grained PCPs:** Understanding the exact constants in the PCP parameters

9 Conclusion

Interactive proofs and PCPs reveal deep connections between proof verification, randomness, and approximation. The ability to verify proofs locally—reading only constant bits—seems almost paradoxical, yet the PCP Theorem shows this is possible for all of NP. These results have revolutionized both complexity theory and our understanding of approximation algorithms, showing that for many optimization problems, even approximate solutions are computationally hard.