Pseudodeterminism for Queries and Communication

Rahul Santhanam

(University of Oxford)

Search Problems

- Search problem $R \subseteq \{0,1\}^* \times \{0,1\}^*$
  - Input: $x \in \{0,1\}^*$
  - Output: y such that $R(x,y)$ holds
- We will only consider polynomially bounded relations R
  - There is c such that for all x,y, $R(x,y)$ holds $\Rightarrow |y| \leq c|x|^c$
- By default, R will be total, i.e, $\forall x \exists y\ R(x,y)$

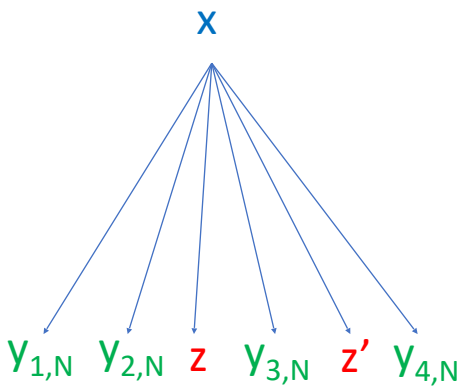
Randomized Algorithms for Search Problems

- Given a search problem R and a parameter ε, a randomized algorithm A solves R with error at most ε if
  - Pr(R(x,A(x)) holds) ≥ 1-ε
- Note that on different random choices of A, different solutions y to R may be output
- How about if we demand a fixed *canonical* output with high probability, with the randomness of A being used only to make A more efficient?
  - "Pseudodeterministic": randomized algorithm with essentially deterministic output

Deterministic, Randomized

Algorithm A for search problem R

Deterministic
(correct solution)

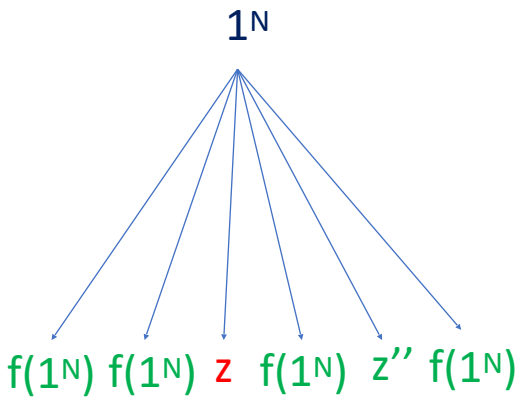


Randomized
(whp, correct solution)

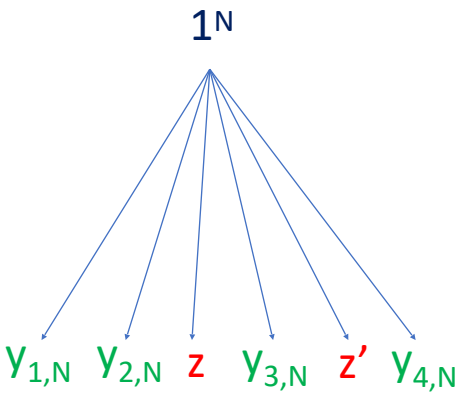
Deterministic, Pseudodeterministic, Randomized

Algorithm A for search problem R

Deterministic
(correct solution)



Pseudodeterministic
(whp, *fixed* correct solution)



Randomized
(whp, correct solution)

Some History

- Pseudodeterminism was defined and studied in [GG11], motivated by applications in cryptography and distributed computing
- Several precursors/independent works
  - [HNOS96]: Single-valued nondeterminism
  - [CLLORS95]: Pseudodeterminism for sampling and communication
  - [HN11]: Pseudodeterministic communication for search problems
- Many subsequent lines of work, on algorithms, concrete complexity, learning and replicability

Unique vs Canonical

- Uniquely solvable search problem $R \subseteq \{0,1\}^* \times \{0,1\}^*$
  - $\forall x \; \exists! y \; R(x,y)$
- Note that any randomized algorithm for R must be pseudodeterministic
- We are typically interested in situations where there are *many* possible solutions for R, and the algorithm needs to identify some *canonical* solution
  - Pseudodeterminism is a property of the algorithm rather than of the problem

Deterministic, Pseudodeterministic, Randomized

- R has efficient deterministic algorithm => R has efficient pseudodeterministic algorithm => R has efficient randomized algorithm
- Are there examples where
  - R is efficiently solvable with randomness but not pseudodeterministically?
  - R is efficiently solvable pseudodeterministically but not deterministically?

First Example: Easy for Randomized, Hard for Pseudodeterministic

- We need the notion of Kolmogorov complexity
- $K(x) = \min \{|p|: U(p, \varepsilon) = x\}$ (where U is universal Turing machine)
- Easy to see that
  - There is constant c such that $K(x) \leq |x| + c$ for all x
  - For at least a $7/8$ fraction of x of length n, $K(x) \geq n - 3$
- Define the search problem $R = \{(1^n, x): |x| = n \ \& \ K(x) \geq n - 3\}$

First Example: Easy for Randomized, Hard for Pseudodeterministic

- Define the search problem $R = \{(1^n, x): |x| = n \ \& \ K(x) \geq n - 3\}$
- There is a linear-time randomized algorithm that solves R with error at most $1/8$: simply output a random string x of length n!
- Proposition: There is no pseudodeterministic algorithm A that solves R with error $< 1/2$ and halts on all computation paths
- Proof: Suppose there exists such an A which is correct on input 1^n for infinitely many n. By exhaustive search over computation paths of A, we can compute the unique y such that $A(1^n)$ outputs y with probability $> 1/2$. But this implies $K(y) \leq \log(n) + O(1)$ – contradiction to correctness for large enough n!

First Example: Easy for Randomized, Hard for Pseudodeterministic

- Define the search problem $R = \{(1^n, x): |x| = n \ \& \ K(x) \geq n - 3\}$
- Crucial feature of argument: it is uncomputable to check if a solution to the search problem is correct

Second Example: Easy for Pseudodeterministic, Hard (?) for Deterministic

- We consider the natural search problem of finding a non-root of a non-zero polynomial p over large fields.
- Let $\{F_n\}_n$ be a sequence of fields such that $n < |F_n| = \text{poly}(n)$ for all n.
- Define the search problem $R = \{(<1^n, C>, \mathbf{a})$: C computes a non-zero polynomial of degree at most n on n variables over F_n, and $C(\mathbf{a}) \neq 0$, where $\mathbf{a} \in F_n{}^n\}$
- R is not total, but we only care about what the algorithm does when solutions exist
- If we could solve R in deterministic polynomial time, we would obtain breakthrough circuit lower bounds!

Second Example: Easy for Pseudodeterministic, Hard (?) for Deterministic

- Define the search problem $R = \{(<1^n, C>, \mathbf{a})$: C computes a non-zero polynomial p of degree at most n on n variables over $F_{n,}$ and $p(\mathbf{a}) \neq 0$, where $\mathbf{a} \in F_n^n\}$
- Schwartz-Zippel Lemma: Suppose p is a non-zero polynomial on n variables of degree d over F_n, and $S \subseteq Fn$. Then for **a** chosen uniformly in S^n, the probability that $p(\mathbf{a}) = 0$ is at most $d/|S|$
- We will use iteratively identify a non-root as follows:
  - Suppose we have already identified $a_1, a_2 \ldots a_i$ in S such that there is a non-root of p with the first i variables set to these values in order, and let p_i be the resulting polynomial on n-i variables
  - For each b in $F_{n,}$ check if $p_i(b, x_{i+2}, \ldots x_n)$ is identically zero using the Schwartz-Zippel test, using repetition to get low error probability. Set a_{i+1} to be the first b for which this is not the case

Second Example: Easy for Pseudodeterministic, Hard (?) for Deterministic

- Define the search problem $R = \{(<1^n, C>, \mathbf{a})$: C computes a non-zero polynomial p of degree at most n on n variables over F_n, and $p(\mathbf{a}) \neq 0$, where $\mathbf{a} \in F_n^n\}$
- We will use iteratively identify a non-root as follows:
  - Suppose we have already identified $a_1, a_2 \ldots a_i$ in S such that there is a non-root of p with the first i variables set to these values in order, and let p_i be the resulting polynomial on n-i variables
  - For each b in F_n, check if $p_i(b, x_{i+2}, \ldots x_n)$ is identically zero using the Schwartz-Zippel test, using repetition to get error probability $< 1/qn^2$, where $q = |F_n|$. Set a_{i+1} to be the first b for which this is not the case
- Since every application of the Schwartz-Zippel test has error probability $< 1/n^3$, this algorithm will output the lex first non-root with probability at least 1-1/n

Second Example: Easy for Pseudodeterministic, Hard (?) for Deterministic

- Define the search problem $R = \{(<1^n, C>, \mathbf{a})$: C computes a non-zero polynomial p of degree at most n on n variables over F_n, and $p(\mathbf{a}) \neq 0$, where $\mathbf{a} \in F_n^n\}$
- Since every application of the Schwartz-Zippel test has error probability $< 1/n^3$, this algorithm will output the lex first non-root with probability at least $1-1/n$
- Underlying phenomenon: we are reducing the search problem to a *decision problem* in BPP

A Couple of Facts

- Proposition (Search vs Decision): A search problem R has an efficient pseudodeterministic algorithm with error $< 1/3$ iff R efficiently reduces to a problem in BPP
- Proposition (Error Reduction) : A search problem R has efficient pseudodeterministic algorithm with error < 0.49 iff R has efficient pseudodeterministic algorithm with error $< 2^{-n}$

White Box, Black Box

- Thus far, we have studied pseudodeterminism in the *white box* setting, where efficiency is measured by computation time
- Understanding the relationship with determinism and randomness in this setting is challenging because of long-standing open questions in derandomization and complexity lower bounds

White Box, Black Box

- Consider an NP search problem R, i.e., R is verifiable in polynomial time
- Under standard derandomization assumptions, if R has efficient randomized algorithm A, it also has efficient deterministic algorithm
  - Enumerate over pseudorandom choices for A and output first valid solution, using verifiability of R to check validity
- Conversely, an efficient deterministic algorithm for the Non-Root problem would have new lower bound implications

White Box, Black Box

- We turn to the *black box* setting, where there are more techniques to show unconditional separations
- Query setting: We study the number of queries to the input needed to solve a search problem R
- Communication setting: We study the amount of communication needed to solve a search problem R, where the input is split between the two parties

Pseudodeterministic Query Complexity

- Given a search problem $R \subseteq \{0,1\}^* \times \{0,1\}^*$
  - We say R has query complexity k(n) [det(R) ≤ k(n)] if there is a deterministic algorithm A which solves R and makes at most k(n) queries to its input for any input of length n
  - We say R has pseudodeterministic query complexity k(n) [pdet(R) ≤ k(n)] if there is a pseudodeterministic algorithm A which solves R with error at most 1/3 and makes at most k(n) queries to any input of length n
  - We say R has randomized query complexity k(n) [rand(R) ≤ k(n)] if there is a randomized algorithm A (i.e., a distribution over deterministic algorithms) which solves R with error at most 1/3 and making at most k(n) queries
- We have rand(R) ≤ pdet(R) ≤ det(R)
  - What are the best separations we can achieve?

Pseudodeterminism vs Determinism

- Nisan showed that for any Boolean function f, $\mathrm{rand}(f) \leq \mathrm{det}(f)^3$
- This simulation result extends to multi-output functions f
- Note that any pseudodeterministic query algorithm solving a total search problem R is a randomized query algorithm for the multi-output function f which maps x to the canonical solution for x
- Hence we have $\mathrm{pdet}(R) = \mathrm{rand}(f) \leq \mathrm{det}(f)^3 \leq \mathrm{det}(R)^3$
- Thus the separation between pseudodeterminism and determinism is at most cubic in the black-box setting, in contrast to white-box setting

Pseudodeterminism vs Randomness

- We consider the search problem Approx-Hamming = $\{(x,v): |wt(x)-v| \leq |x|/10\}$
- Simple and efficient randomized algorithm for Approx-Hamming
  - Sample $O(1)$ locations of x uniformly at random and output the fraction of 1s
- Clearly Approx-Hamming requires $\Omega(n)$ queries deterministically
  - If we output a value v after fewer than $0.8n$ queries, v can't be correct both for the input where unqueried bits are all 1 and input where all unqueried bits are 0
- Pseudodeterministic query complexity of Approx-Hamming is $\Omega(n^{1/3})$ by previous slide, but can we show a better lower bound?

Pseudodeterminism and Sensitivity

- We say that a function f has sensitivity k at input x if there are k locations i such that $f(x) \neq f(x^i)$, where x^i is x with the i'th bit flipped
- $Sens(f)$ = max of sensitivity of f over all inputs
- We say that a function f is consistent with search problem R if for all x, $R(x, f(x))$ holds
- Lemma: If every function f consistent with R has sensitivity at least k, then $pdet(R) \geq k/3$

Pseudodeterminism and Sensitivity

- Lemma: If every function f consistent with R has sensitivity at least k, then $pdet(R) \geq k/3$
- Consider any pseudodet algorithm A for R. This yields some function f consistent with R, i.e., the f producing a canonical solution for A. Consider an input x for f with sensitivity at least k
- For each location i where f is sensitive at x, A must query i with probability at least $1/3$
  - A outputs some value v on x with probability at least $2/3$ and some other value v' on x with probability at least $2/3$
- Hence the expected number of queries made by A is at least $k/3$

A Separation for Approx-Hamming

- Theorem: pdet(Approx-Hamming) $\geq n/6$
- Lemma: For every f consistent with Approx-Hamming, Sens(f) $\geq n/2$
- We define a sequence of inputs $x_0, x_1, \ldots, x_{n/2}$, where x_i has weight i for each i
- For each $i \geq 0$, we maintain the invariant that either f has sensitivity $\geq n/2$ at x_i, or $f(x_i) = f(x_{i+1})$
  - If f does not have sensitivity $\geq n/2$ at x_i, then there is some neighbour x_{i+1} of x_i with weight i+1 such that $f(x_i) = f(x_{i+1})$, as there are $n-i \geq n/2$ neighbours of x_i with weight i+1
- Either we get a sensitive input, or we get a contradiction to consistency of f, as $f(x_0) = f(x_{n/2})$ cannot 0.1-approximate both 0 and n/2

Pseudodeterminism for TFNP Problems

- Approx-Hamming gives an optimal separation between randomized query complexity and pseudodeterministic query complexity
- However solutions to Approx-Hamming are not *efficiently verifiable*
  - Given a candidate solution v, we need to query $\Omega(n)$ bits of x to verify that (x,v) is in Approx-Hamming
- The analogue of TFNP in the query world is the class of total search problems for which we can verify a solution by making at most polylog(n) queries
- Are there strong separations between pseudodeterminism and randomness for $\text{TFNP}^{\text{query}}$?

The Find1 Problem

- We study the very natural promise search problem Find1
- Find1 = {(x,i): x has at least n/2 1s, and $x_i = 1$}
  - We are only required to solve Find1 when x satisfies the promise that at least half of its bits are 1
- Note that there is a trivial randomized algorithm for Find1 with query complexity O(1): query O(1) random locations and output the smallest location at which there is a 1, if any
- Also trivial deterministic lower bound of n/2, as adversary can set all queried bits to 0
- Open Question: Is pdet(Find1) = Ω(n)?

A Lower Bound for Find1

- We study the very natural promise search problem Find1
- Find1 = {(x,i): x has at least n/2 1s, and $x_i = 1$}
  - We are only required to solve Find1 when x satisfies the promise that at least half of its bits are 1
- Proposition: Find1 is complete for NP^{query} promise search problems with small randomized query complexity
  - Any pseudodeterministic query lower bound for an NP^{query} promise search problem with small randomized query complexity transfers to Find1
- Theorem: There is a $TFNP^{query}$ search problem R with rand(R) = O(1) and pseudodet(R) = $\Omega(n^{1/2})$
- Corollary: pseudodet(Find1) = $\Omega(n^{1/2})$

A Lower Bound for $\mathrm{TFNP}^{\mathrm{query}}$

- Theorem: There is a $\mathrm{TFNP}^{\mathrm{query}}$ search problem R with $\mathrm{rand}(R) = O(1)$ and $\mathrm{pseudodet}(R) = \Omega(n^{1/2})$
- We consider the Falsified-Clause-Search problem $\mathrm{FCS}(\phi)$ for a randomly chosen unsatisfiable d-CNF ϕ on n variables with cn clauses for constant d and large enough constant c
- $\mathrm{rand}(\mathrm{FCS}(\phi)) = O(1)$
  - For randomly chosen ϕ, every assignment x falsifies a constant fraction of clauses
- $\mathrm{pseudodet}(\mathrm{FCS}(\phi)) = \Omega(n^{1/2})$

A Lower Bound for $TFNP^{query}$

- Lemma: pseudodet(FCS(ϕ)) = $\Omega(n^{1/2})$
- Proof
  - Let A be the best pseudodeterministic algorithm for FCS(ϕ) and f the canonical function corresponding to it
  - [G98,AR01,BGIP01] NS(ϕ) = Ω(n)
  - Lemma: deg(f) ≥ NS(ϕ) = Ω(n)
  - [H19] Sens(f) ≥ $deg(f)^{1/2}$ = $\Omega(n^{1/2})$
  - pseudodet (FCS(ϕ)) ≥ Sens(f) = $\Omega(n^{1/2})$

A Natural Combinatorial Problem

- Call an n-coloring C of the vertices of the n-dimensional hypercube (excluding the all-zeroes vertex) admissible if it satisfies the following condition
  - $x_{C(x)} = 1$ for all x
- Conjecture: For every admissible n-coloring of the n-dimensional hypercube, there is a vertex v of weight $n/2$ such that $\Omega(n)$ of the neighbours of v have a different color than v
- If Conjecture holds, then pseudodet(Find1) = $\Omega(n)$
- What is known: there is v of weight $n/2$ such that $\Omega(n^{1/2})$ of its neighbours are differently colored

Pseudodeterminism for Communication Complexity

- We consider the 2-party setting, where Alice and Bob communicate to compute a function f(x,y), where Alice has x and Bob has y
- Search problem $R \subseteq \{0,1\}^* \times \{0,1\}^* \times \{0,1\}^*$
  - Input: x, y $\in \{0,1\}^*$
  - Output: z such that R(x,y,z) holds
- Communication complexity measures rand, pseudodet, det can be defined in analogy to the corresponding query measures
- Recall: det and pseudodet are polynomially related in the query setting, and there are exponential separations between pseudodet and rand
- Communication setting is qualitatively different

An Exponential Separation between Determinism and Pseudodeterminism

- Consider the search problem UNEQUAL = $\{(x,y,i): x_i \neq y_i\}$
- Using a fooling set argument, it can be shown that det(UNEQUAL) = $\Omega(n)$
- Proposition: pseudodet(UNEQUAL) = O(log(n) loglog(n))
- Proof: Alice and Bob communicate to find the lex first i such that $x_i \neq y_i$. To do this, they use binary search, where each level of the recursion involves one call to the Equality protocol, which can be implemented with cost O(loglog(n)) for error 1/10log(n). The bound follows because there are log(n) levels of recursion.

Pseudodeterminism vs Randomness

- Exponential separations between pseudodeterministic and randomized communication complexity have turned out to be very hard to achieve
- Candidate search problem: Approx-Hamming-Dist =

  $\{(x,y,v): |\Delta(x,y) - v| \leq 0.1n\}$
- As with Approx-Hamming in the query model, there is a simple constant-cost randomized protocol for Approx-Hamming-Dist
- Conjecture: pseudodet(Approx-Hamming-Dist) = $\Omega(n)$
- Very recently, [GHRSSY26] obtained an exponential separation for a related partial search problem

Open Problems

- Optimal pseudodeterministic query lower bound for FIND1
- Optimal pseudodeterministic communication lower bound for Approx-Hamming-Dist
- Exponential separation between randomized and pseudodeterministic complexity for a communication problem in $\mathrm{TFNP}^{\mathrm{cc}}$