Randomness in Complexity: Introduction and Linearity test

Michel de Rougemont
IRIF-CNRS

Abstract

We study both randomized algorithms, randomized complexity classes and randomized data. An introduction to Testers, streaming algorithms and PCP. Communication complexity is a basic tool to prove lower bounds for both Testers and streaming algorithms.

Contents

1 Introduction

We study both randomized algorithms, randomized complexity classes and data which follow specific distributions. One of the goals of the course is to present recent developments to go *beyond the worst case analysis* of algorithms.

- We will first study Testers, Streaming and Online algorithms,
- We review the BPP(*Bounded Probabilistic* P) , IP, PCP complexity classes and introduce the *Communication Complexity* classes,
- We then consider Online algorithms beyond the worst-case such as Facility location, Online paging and sparse recovery.

Consider a typical weighted matching problem in a graph. We could get the graph as a stream of edges and ask:

- Is there a matching of weight greater than c,
- A near-optimal matching
- A stable near-optimal matching

For a general problem, we can ask similar questions, and generalize them for a distribution μ. These are the algorithmic questions we address. We insist on approximate solutions, both for decision and search problems. The program of the 8 sessions is:

1. Introduction, BLR linear tester
2. Communication Complexity
3. Testers on words, trees and graphs, lower bounds
4. Streaming algorithms and streaming testers, lower bounds, Online Bipartite matching
5. Random order model for secretary and facility location problems
6. Instance optimality, Auctions
7. Online paging
8. Sparse recovery

2 Classical Probabilistic complexity classes

A probabilistic algorithm is a constructive procedure which uses a new instruction : *random choice*. On can flip a coin or randomy choose a value among k equiprobable distinct values. We consider in what follows a random choice as the selection of the values 0 or 1 with probability $\frac{1}{2}$. We define the notion of probabilistic acceptance of a language, through the use of specific non deterministic Turing machines.

Definition 1 *A* **probabilistic Turing machine** *is a non deterministic Turing machine whose computation tree is a complete binary tree, i.e. all the branches have the same length.*

A *probabilistic execution* starts with the initial description $(q_0.x)$ and follows at each step a possible transition of the machine. At each non deterministic branch, the machine makes a *random choice* between two possible transitions with an uniform distribution. The computation tree is a complete binary tree of depth t where all the paths have equal probability. The probabilistic space is $\Omega_t = \{(\rho, \frac{1}{2^t}) : \rho \in \{0,1\}^t\}$.

The notion of acceptance is defined by comparing the number of accepting paths $acc_M(x)$ and rejecting paths $rej_M(x)$. In a equivalent way, we talk of acceptance probability as the quotient of the number of accepting paths by the total number of paths.

$$\mathbb{P}rob_\Omega[M \text{ accepts x}] = \frac{acc_M(x)}{2^t}$$
$$\mathbb{P}rob_\Omega[M \text{ rejects x}] = \frac{rej_M(x)}{2^t}$$

A **run** or **experiment** is a path in the computation tree leading to an accepting, or rejecting state.

2.1 Probabilistic versions of the class P

1. PP (*probabilistic* P)
2. RP (*Random* P)
3. BPP(*Bounded Probabilistic* P)

Definition 2 *The class PP is the class of languages $\mathcal{L}$ for which there is a probabilistic polynomial time bounded Turing machine such that :*

- *if $x \in \mathcal{L}$, then $\mathbb{P}rob_\Omega[M$ accepts x$] > \frac{1}{2}$,*
- *if $x \notin \mathcal{L}$, then $\mathbb{P}rob_\Omega[M$ accepts x$] \leq \frac{1}{2}$.*

An important property of the class PP is:

Proposition 1 $NP \subseteq PP$

Proof: Let a language $\mathcal{L} \in NP$, defined by a machine M. Let us transform M into M' such that the computation tree is complete and such that M accepts x iff M' accepts x. Let us define a new machine M" whose depth is the depth of the machine M plus one. On the first left branch, we duplicate the computation tree of M' and on the right first branch we construct a complete binary tree of accepting states only (see figure below).

M *accepts* x iff M' *accepts* x iff $\mathbb{P}rob_\Omega[M$" accepts x$] > \frac{1}{2}$. One concludes that $\mathcal{L} \in PP$. $\square$

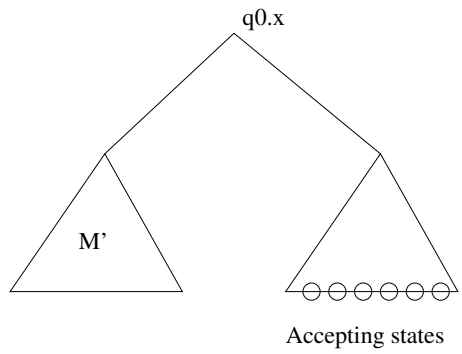

Figure 1: The computation tree of M" which accepts with probability $> \frac{1}{2}$ iff $x \in \mathcal{L}$

The following two classes RP and BPP are more interesting from a practical and algorithmic point of view.

Definition 3 *RP is the class of languages $\mathcal{L}$ for which there is probabilistic polynomial time bounded Turing machine M such that:*

- *if $x \in \mathcal{L}$, then $\mathbb{P}rob_\Omega[M$ accept x$] > \frac{1}{2}$*
- *if $x \notin \mathcal{L}$, then $\mathbb{P}rob_\Omega[M$ accept x$] = 0$*

Notice the asymmetry of this definition with regard to $x \in L$. A language is in the class co-RP if its complement is in the class RP. Another important class is the class $ZPP = RP \cap co-RP$. If a problem is in ZPP, one can run on an input x an RP algorithm and independently a co-RP algorithm : with probability greater than 1/2, the two algorithms will give coherent answers. If the answers are incoherent, it is then difficult to detect the right answer. If we repeat this computation k time, the probability to obtain incoherent answers is lower than $(1/2)^k$. One obtains then a probabilistic algorithm without error whose time complexity is only expected to be polynomial. Such algorithms are also called *Las Vegas* algorithms. A more general model of *Las Vegas* algorithms does not presuppose the computation tree to be complete.

Definition 4 *BPP is the class of languages $\mathcal{L}$ for which there is probabilistic Turing machine working in polynomial time and a constant $\varepsilon > 0$ such that:*

- *if* $x \in \mathcal{L}$, *then* $\mathbb{P}rob_{\Omega}[M$ accepts x$] \geq \frac{1}{2} + \varepsilon$,
- *if* $x \notin \mathcal{L}$, *then* $\mathbb{P}rob_{\Omega}[M$ accepts x$] \leq \frac{1}{2} - \varepsilon$.

Notice that ε must be constant, independent of the input x. The class BPP is also known as the class of problems which can be solved by by *Monte Carlo* algorithms. A fundamental difference exists between the classes PP$andBPP$. Let x be an input of size n, and $m = 2^n$ the number of leaves of the computation tree. If the number of accepting leaves is $\frac{m}{2} + 1$, then x is accepted is the PP sense. In order to be accepted in the BPP sense, there must be an ε, independent of x, such that the number of accepting leaves is greater than in $m(\frac{1}{2}+\varepsilon)$.

Example: Consider the computation tree for $m = 16$, $\varepsilon = 0.09$ and 9 accepting leaves. The number of accepting leaves (9) is not greater than $16/2+16*0,09 = 9,44$ and the input is accepted in the PP sense but not in the BPP sense.

A fundamental property of the class BPP we now explain, is that the error probability of $(1/2) - \varepsilon$ can be made arbitrarly small between 0 and $\frac{1}{2}$ and taking a majority decision, in particular very close to 0.

2.2 Probabilistic Verification

The class NP is often called the class of problems *verifiable in polynomial time*. Indeed, a language L is in the class NP iff there exists a binary relation R which satisfies:

- R is decidable in polynomial time;
- there exists k such that if $R(x, y)$, then $|y| < O(|x|^k)$;
- $L = \{x : \text{there exists } y \text{ such that } R(x, y)\}$.

We say that y is a *witness* or a *proof* of the fact that $x \in L$. This situation can be represented as follows: a prover P transmits a proof y to a verifier V to convince him that $x \in L$. The verifier accepts x iff $R(x, y)$ is satisfied and this can be computed in polynomial time.

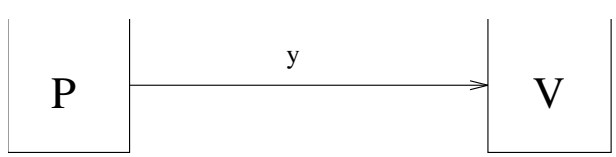

Figure 2: The classical schema between a prover P and a verifier V.

The classes introduced in the previous chapter are associated with decision problems and generalize the classic complexity classes. We wish to generalize NP as we generalized P to BPP. **Probabilistic verification** is the problem of verifying with a probabilistic algorithm whether two values a, b are such that $f(a) = b$ for a given function f. The verifier will follow a procedure which uses both randomness and interaction with a prover, to test whether the prover knows that $f(a) = b$. The probabilistic verification generalizes the previous schema in two specific ways : the verifier will use randomness and interaction, i.e. will determine new questions to the prover. The objective is to show that some problems which have long proofs (of exponential length) can be verified in polynomial time with high probability. This new notion modifies the classical notion of a proof in logic.

2.2.1 Interactive proofs

The previous schema is generalized when the verifier interacts with the prover and follows a BPP algorithm. We consider a model where the random bits are secret, i.e. not known to the prover.

Let P and V be two Turing machines which communicate through a common tape where they exchange messages. The machine V uses a secret[1]

random coin and follows a BPP algorithm, whereas the prover P has no constraints[2] The input x is known of both P and V and the two machines exchange messages on a common tape. The exchanges are described by a sequence of functions $V_\rho^1, .., V_\rho^k$ which define the messages generated by V (also called *questions to the prover*) and a sequence of functions $P^1, ..., P^{k-1}$ which define the messages generated by P (also called *the answers of the prover*).

The verifier accepts or rejects in polynomial time: one writes $P.V(x) = 1$ if V accepts and $P.V(x) = 0$ if V rejects. Let x be an input of length n and $\rho \in \{0,1\}^m$ a random word representing the random bits generated by V.

A **protocol** is a sequence of functions $V_\rho^1, ..., V_\rho^{k-1}$ and of functions $P^1, ..., P^{k-1}$. Each V_ρ^j is a function of $(\{0,1\}^*)^j$ into $\{0,1\}^*$ for $0 < j < k$ where $k \leq n^p$ is computable in polynomial time. Each P_ρ^j is a function of $(\{0,1\}^*)^{j+1}$ into $\{0,1\}^*$. The function V_ρ^k maps $(\{0,1\}^*)^k$ into $\{0,1\}$. No restriction is made on the complexity of the P_ρ^j functions. We write:

$$\begin{array}{lcl} V \text{ computes } u_1 & = & V_\rho^1(x) \\ P \text{ computes } v_1 & = & P^1(x, u_1) \\ V \text{ computes } u_2 & = & V_\rho^2(x, v_1) \\ P \text{ computes } v_2 & = & P^2(x, u_1, u_2) \end{array}$$

[1]The prover ignores the result of the coin flipping. In another model, the *AM* (Arthur–Merlin) games, the coin flipping is public, i.e. known to the prover. The two models are equivalent.

[2]The prover can in principle use non-recursive oracles. A prover limited within the class PSPACE would give, however, an equivalent model.

...

P computes $v_{k-1} \;=\; P^{k-1}(x, u_1, u_2, ..., u_{k-1})$
V computes $u_k \;=\; V_\rho^k(x, v_1, ..., v_{k-1}) \in \{0, 1\}$

The functions V_ρ^i depend on the random vector ρ and the last value u_k is the decision of the verifier V. The result $u_k = 1$ (resp. $v_k = 0$) is also written $P.V(x) \;=\; 1$ (resp. $P.V(x) \;=\; 0$). We consider the following probability:

$$\mathbb{P}rob_\rho[P.V(x) \;=\; 1]$$

where the probabilistic space is the set of random boolean sequences of length m where m is $O(n^r)$, all equiprobable.

Definition 5 *A language L admits an interactive proof if there is a protocol such that the verifier follows a BPP algorithm and for all x :*

- *if $x \in L$, there exists a prover P such that:*

$$\mathbb{P}rob[P.V(x) = 1] = 1$$

- *if $x \notin L$, then for any prover P':*

$$\mathbb{P}rob[P'.V(x) = 1] \leq \frac{1}{2}$$

Notice that the first condition assumes no error. In a more general definition, we could allow an error in both conditions and take $\mathbb{P}rob[P.V(x) = 1] \geq 1 - \varepsilon$. The first condition is sometimes called the *completeness* and the second condition is called the *soundness*. An *honest* prover always satisfies the first condition.

Definition 6 *The class IP is the class of languages L for which there is an interactive proof.*

Notice that $\mathsf{NP} \subseteq IP$, because the protocol is reduced to a single interaction. In the case of SAT, the verifier asks the prover for a valuation which satisfies all the clauses and then verifies it in polynomial time. In the same way $coRP \subseteq IP$. The first indication that IP is a class much larger than NP is the existence of protocols for problems in coNP. The most classical example is the protocol for the non-isomorphism of graphs, described in the appendix. The protocol for the verification of the permanent introduces fundamental algebraic techniques and also uses Schwartz's lemma. The protocol for QBF is a generalization of the protocol for the permanent.

2.2.2 Probabilistic checkable proofs: PCP

In an IP protocol, the prover is defined by a sequence of arbitrary functions which define the answers to the questions of the verifier. The prover can answer differently the same question at two different stages. Hence the prover is not an oracle which always answers identically the same questions.

We now restrict the prover by forcing him to write in advance a proof π on a tape which can be read by the verifier with the following mechanism. Given an address i, the prover has to answer $\pi(i)$, the ith bit of the proof π. Notice that the proof π can be long of exponential length but is only tested on a polynomial number of addresses.

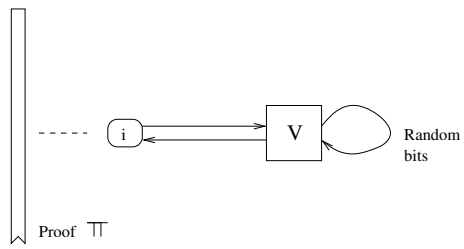

Figure 3: The interaction between a proof π and the verifier V in the PCP model.

A PCP **verifier** is a probabilistic Turing machine M with two specific tapes: one tape for the random bits and another for the proof, also called the oracle tape. Given a word k interpreted as an integer i in binary, the machine M accesses the symbol of the proof π at the address i. The questions of V do not depend on the answers to the questions to the oracle.

A verifier is in the class $\mathsf{PCP}(r(n), q(n))$ if it uses $O(r(\mid x \mid))$ random bits and $O(q(\mid x \mid))$ questions to the oracle tape.

Definition 7 *A* $\mathsf{PCP}(r(n), q(n))$ **verifier** *is defined by two functions:*

- $Q_V(x, \rho) = (w_1, ..., w_q)$ *where* ρ *is a random vector of length* $O(r(\mid x \mid))$*,* w_i *is the address of the* i*th question,* $R_i = \pi(w_i) \in \{0, 1\}$ *is the bit of* π *at address* w_i*, and* q *is* $O(q(\mid x \mid))$*.*
- $F(x, \rho, R_1, ..., R_q) \in \{0, 1\}$ *is the decision to accept or to refuse.*

We write $V(x, \rho, \pi) = 1$ if $F(x, \rho, R_1, ..., R_q) = 1$ and V accepts. Similarly, $V(x, \rho, \pi) = 0$ if $F(x, \rho, R_1, ..., R_q) = 0$ and V rejects.

Definition 8 *A language* L *is in the class* $\mathsf{PCP}(r(n), q(n))$ *or admits a verifier* $\mathsf{PCP}(r(n), q(n))$ *if there exists a verifier* V *of the class* BPP *such that for all* x*:*

- *if* $x \in L$*, there exists a proof* π *such that* $\mathbb{P}rob[V(x, \rho, \pi) = 1] = 1$*,*

- *if $x \notin L$, for any proof π',*

$$\mathbb{P}rob[V(x, \rho, \pi') = 1] \leq \frac{1}{2}$$

Observe the following equivalences:

- $\mathsf{PCP}(0, n^k) = \mathsf{NP}$ because the verifier is deterministic and reads a proof of polynomial length.
- $\mathsf{PCP}(n^k, 0) = coRP$ because the machine is probabilistic and makes mistakes only in case $x \notin L$.

A partial result asserts that $\mathsf{NP} = \mathsf{PCP}(\log n, \log n)$ [2] but the fundamental result [1] is:

$$\mathsf{NP} = \mathsf{PCP}(\log n, 1)$$

i.e. the verifier needs $\log n$ bits and a constant number of questions to capture NP or to check proofs of polynomial length. In this model the verifier can check whether a proof π is correct by observing a very small proportion of the proof. Such proofs are called **transparent** or **holographic**.

Example. Consider two proofs of the assertion *n is prime*: the first is a classic proof while the second is holographic but of exponential length.

Classical proof: consider a table of size $n^{\frac{1}{2}}$ lines where every line quotes the division of n by i, and we write: $n = q.i + r_i$. To say that n is prime, is equivalent to say that the rest r_i of the division of n by i for $i = 2, ..., n/2$ is strictly positive. All the lines are important because if n is not prime two lines are 0 and a random choice of a line has only an exponentially small chance to find the correct i. This proof is not transparent.

Transparent proof: let us show how to generate a transparent proof, using the Solovay–Strassen test.

If n is prime, then for any $a < n$, $(\frac{a}{n}) = a^{\frac{n-1}{2}} \ (n)$. If n is composite, then with a probability greater than 1/2, either $pgcd(a, n) > 1$ or $(\frac{a}{n}) \neq a^{\frac{n-1}{2}}$ when a is random between 1 and $n - 1$. Let us generate n lines.

$$pgcd(i, n) = 1 \ \ and \ \ (\frac{i}{n}) = i^{\frac{n-1}{2}} \ (n)$$

If n is not prime, then for more than half of the lines i, the assertion of line i is false. If we sample the lines, we will find such an i with high probability and this proof is transparent. Notice that these two proofs are of exponential size.

In the appendix, we show that $\mathsf{NP} \subseteq \mathsf{PCP}(n^3, 1)$ by giving a protocol for the $\mathsf{3SAT}$ problem , leaving some of the justifications as exercises.

The class IP was defined in 1985 in two different ways. Golwasser, Micali and Rackoff [11] introduced *interactive proofs* where the random bits of the verifier are secret while Babai [3] introduced the *Arthur-Merlin* games where Arthur is the verifier, Merlin the prover and Arthur's random bits are public. These games generalize the *games against the nature* introduced by Papadimitriou [15]. The generalization to several provers was introduced in [5] and the PCP model was developed in [2] and [1]. Additional references are [4] and [12].

3 Testers and Streaming Algorithms

Given a distance between inputs, an ε-tester for a property P accepts all inputs which satisfy the property and rejects with high probability all inputs which are ε-far from inputs that satisfy the property. Inputs which are ε-close to the property determine a gray area where no guarantees exists. These restrictions allow for sublinear algorithms and even $O(1)$ time algorithms, whose complexity only depends on ε.

Let $\mathbf{K}$ be a class of finite structures with a normalized distance dist between structures, i.e. dist lies in $[0, 1]$. For any $\varepsilon > 0$, we say that $U, U' \in \mathbf{K}$ are *ε-close* if their distance is at most ε. They are *ε-far* if they are not ε-close. In the classical setting, satisfiability is the decision problem whether $U \models P$ for a structure $U \in \mathbf{K}$ and a property $P \subset \mathbf{K}$. A structure $U \in \mathbf{K}$ *ε-satisfies* P, or U is ε-close to $\mathbf{K}$ or $U \models_\varepsilon P$ for short, if U is ε-close to some $U' \in \mathbf{K}$ such that $U' \models P$. We say that U is ε-far from $\mathbf{K}$ or $U \not\models_\varepsilon P$ for short, if U is not ε-close to $\mathbf{K}$.

3.1 Property Testing

Definition 9 (Property Tester [10]) *Let $\varepsilon > 0$. An ε-*tester *for a property $P \subseteq \mathbf{K}$ is a randomized algorithm A such that, for any structure $U \in \mathbf{K}$ as input:*
(1) If $U \models P$, then A accepts;
(2) If $U \not\models_\varepsilon P$, then A rejects with probability at least $2/3$.[3]

A *query* to an input structure U depends on the model for accessing the structure. For a word w, a query asks for the value of $w[i]$, for some i. For a tree T, a query asks for the value of the label of a node i, and potentially for the label of its parent and its j-th successor, for some j. For a graph a query asks if there exists an edge between nodes i and j. We also assume that the algorithm may query the input size. The *query complexity* is

[3]The constant $2/3$ can be replaced by any other constant $0 < \gamma < 1$ by iterating $O(\log(1/\gamma))$ the ε-tester and accepting iff all the executions accept

the number of queries made to the structure. The *time complexity* is the usual definition, where we assume that the following operations are performed in constant time: arithmetic operations, a uniform random choice of an integer from any finite range not larger than the input size, and a query to the input.

Definition 10 *A property $P \subseteq \mathbf{K}$ is* testable*, if there exists a randomized algorithm A such that, for every real $\varepsilon > 0$ as input, $A(\varepsilon)$ is an ε-tester of P whose query and time complexities depend only on ε (and not on the input size).*

The first ingredient in a tester is to define the distance used and the queries allowed. For example, on strings, we can use the Hamming distance, the Edit distance, or some extension such as the Edit distance with moves. A query $Q(i,k)$ on a word w finds the factor $w[i], w[i+1], ...w[i+k]$, i.e. the subword in position i of length k. We will show that any regular property on words in testable for the Edit distance. On graphs, the Edit distance between two graphs is the minimum number of edges necessary to add or remove to have the two graphs isomorphic. The property of being 3-colorable, an NP-complete property is testable.

3.2 The linearity test

Let $f : F_2^n \to F_2$ be a boolean function. Such a function is linear if either property is satisfied:

- $\forall x, y \in F_2^n \quad f(x) + f(y) = f(x+y)$
- $\exists a \in F_2^n \quad f(x) = a.x$, i.e. $f(x) = \sum_{i \in S} x_i$ for $S \subseteq \{1, 2, ...n\}$

Notice that these 2 conditions are equivalent. Clearly the second condition implies the first. To show that the first implies the second, use an induction on n. For $n = 1$, find a by setting $x = 1$. If $f(1) = 1$ then $a = 1$ otherwise $a = 0$. Assume the property true for $n-1$. Find a_n by checking $f(1, 0, ...0)$. Any x can be written as $(1, 0, ...0) + (0, x_2, ...x_n)$ or as $(0, 0, ...0) + (0, x_2, ...x_n)$. We then apply the induction hypothesis and prove the result.

Consider the approximate version of the linearity conditions:

- for most $x, y \in F_2^n \quad f(x) + f(y) = f(x+y)$
- $\exists a \in F_2^n$ such that for most x $f(x) = a.x$, i.e. $f(x) = \sum_{i \in S} x_i$ for $S \subseteq \{1, 2, ...n\}$

It is interesting to note that the second condition implies the first and it is not obvious to prove the opposite, i.e. the first condition implies the second. It is another motivation to introduce the BLR test.

Let K be the class of boolean functions $f : F_2^n \to F_2$ and let $\mathcal{L}$ the class of linear functions. The distance between two functions f, g is the number of inputs on which they

disagree, i.e. $\frac{\#x:f(x)\neq g(x)}{2^n} = Prob[f(x) \neq g(x)]$. Given a function f and a value i, it is natural to ask for $f(i)$ and this is a *query*. Can we efficiently test if $f \in \mathcal{L}$ or if f is ε-far from $\mathcal{L}$?

The answer is "yes", with the following test introduced by Blum, Luby and Rubinfeld in [7]:

BLR Test(f)
Input: $f : F_2^n \to F_2$

1. Generate uniformly and independently $x, y \in F_2^n$,
2. Reject if $f(x) + f(y) \neq f(x + y)$,

Accept.

If $f \in \mathcal{L}$, clearly f passes the Test with probability 1. We have to prove that if f is ε-far to a linear function then $Prob[f$ passes the test$] \leq c$. Equivalently, by contraposition, If $Prob[f$ passes the test$] > c$, then f is ε-close to a linear function. We will show in theorem 1 that we can take $c = 1 - \varepsilon$. It is a hard part of the result and we will use the Fourier Analysis to prove it.

3.2.1 Discrete Fourier Analysis

Let $f : \{-1, +1\}^n \to R$ be a boolean function and let χ_S be the function $\chi_S(x) = \Pi_{i\in S} x_i : \{-1, +1\}^n \to \{-1, +1\}$ the function which computes the parity or XOR of the x_i. A boolean function can also be viewed as a function into $\{-1, +1\}$ where $+$ is replaced by $*$ if we interpret 0 by 1 and 1 by -1. The test $f(x) + f(y) = f(x + y)$ is replaced by the test $f(x) \cdot f(y) = f(x \cdot y)$.

The Fourier expansion of f is:

$$f = \sum_S \widehat{f(S)} \chi_S$$

Example: $Min_2(x_1, x_2) = -1/2 + x_1/2 + x_2/2 + x_1.x_2/2$

3.2.2 Orthonormal Basis

Definition 11 *Let* $< f, g >= \mathbb{E}[f(x).g(x)] = \sum_x f(x).g(x)/2^n$

Lemma 1 $< \chi_S, \chi_T >= 1$ *if* $S = T$ *and* 0 *otherwise.*

Notice that $\widehat{f(S)} =< f, \chi_S >= Prob[\ \ f(x) = \chi_S\ (x)] - Prob[f(x) \neq \chi_S(x)\quad]$.
If $\mathsf{dist}(f, g) = Prob[f(x) \neq g(x)\quad]$. Hence $\widehat{f(S)} = 1 - 2.\mathsf{dist}(f, \chi_S)$.

3.2.3 Parseval equality and convolution

Lemma 2 $\widehat{f(S)} =< f, \chi(S) >= E[f(x).\chi(S)(x)]$

Proof: Apply the definition:

$$f = \sum_S \widehat{f(S)}\chi_S$$

$$< f, \chi(S) >=< \sum_S \widehat{f(S)}\chi_S, \chi_S >=< \widehat{f(S)}\chi_S, \chi_S >= \widehat{f(S)}$$

□

Lemma 3 *(Plancherel)* $< f, g >= \sum_S \widehat{f(S)}.\widehat{g(S)}$

Proof: $< f, g >= \sum_x f(x).g(x)/2^n = E_x[f(x).g(x)] = \sum_{S,T} \widehat{f(S)}.\widehat{g(S)} < \chi_S, \chi_T >= \sum_S \widehat{f(S)}.\widehat{g(S)}$ □

Lemma 4 *(Parseval)* $\sum_S \widehat{f(S)}^2 =< f, f >$

Proof: $< f, f >= \sum_x f(x)^2/2^n = E_x[f(x)^2] = \sum_S \widehat{f(S)}^2$.
□

Let us define the *convolution* $f*g$ of two functions $F_2^n \to R$ as: $f*g(x) = E_y[f(y).g(x\cdot y)]$. Notice that $E_y[f(y).g(x \cdot y)] = E_y[f(x \cdot y).g(y)]$.

Lemma 5 *Let f, g two functions: $F_2^n \to R$. Then, forall S: $\widehat{f * g(S)} = \widehat{f(S)}.\widehat{g(S)}$*

Proof: $\widehat{f * g(S)} = E_x[f * g(x).\chi_S(x)]$
$\widehat{f * g(S)} = E_x[E_y[f(y).g(x \cdot y)].\chi_S(x)]$
Set $x \cdot y = z$. Notice that $\chi_S(x) = \chi_S(y) \cdot \chi_S(z)$ because χ_S is a linear function. Then:
$\widehat{f * g(S)} = E_{y,z}[[f(y).g(z)].\chi_S(y) \cdot \chi_S(z)]$
$\widehat{f * g(S)} = E_{y,z}[f(y).\chi_S(y)].[g(z)].\chi_S(z)]$
$\widehat{f * g(S)} = \widehat{f(S)}.\widehat{g(S)}$ □

3.2.4 BLR Analysis

Theorem 1 *If $Prob[f$ passes the test$] = 1 - \varepsilon$, then f is ε-close to a linear function.*

Proof:

$$1-\varepsilon = Prob[\text{passes the test}] = Prob[f(x\cdot y) = f(x).f(y)]$$

$$1/2 + 1/2.f(x).f(y).f(x\cdot y) = \begin{cases} 1 \; if \;\; f(x\cdot y) = f(x).f(y), \\ 0 \; if \;\; f(x\cdot y) \neq f(x).f(y) \end{cases}$$

$$\begin{aligned} 1-\varepsilon = Prob[\text{passes the test}] &= \mathbb{E}_{x,y}[1/2 + 1/2.f(x).f(y).f(x\cdot y) \\ &= 1/2 + \mathbb{E}_x[1/2.f(x).\,\mathbb{E}_y[f(y).f(x\cdot y)]] \\ &= 1/2 + \mathbb{E}_x[1/2.f(x).(f*f)(x)] \\ &= 1/2 + 1/2\textstyle\sum_S \widehat{f(S)}.\widehat{(f*f)}(S) \text{ (Plancherel lemma 3)} \\ &= 1/2 + 1/2\textstyle\sum_S \widehat{f(S)}^3 \text{ (lemma 5) Then:} \end{aligned}$$

$$1-2.\varepsilon = \sum_S \widehat{f(S)}^3)$$

$$1-2.\varepsilon \leq max_S \widehat{f(S)}.\sum_S \widehat{f(S)}^2$$

$$1-2.\varepsilon \leq max_S \widehat{f(S)} \text{ using Parseval's inequality}$$

Recall that $\widehat{f(S)} = 1 - 2.\mathsf{dist}(f, \chi_S)$. Hence:

$$1-2.\varepsilon \leq max_S \widehat{f(S)}$$

Hence there exists S_* such that:

$$1-2.\varepsilon \leq \widehat{f(S_*)}$$

$$1-2.\varepsilon \leq 1 - 2.\mathsf{dist}(f, \chi_{S_*})$$

$$\mathsf{dist}(f, \chi_{S_*}) \leq \varepsilon$$

□

We conclude that f is close to the linear function χ_{S_*}. At this point, we do not know S_*. We can however conceive the following corrector:

Corrector(f)
Input: $f : F_2^n \to \{-1, +1\}, x \in F_2^n$

1. Choose $y \in_r F_2^n$,
2. Output $f(y).f(x+y)$.

Lemma 6 *Assume $f : F_2^n \to \{-1, +1\}$ is ε-close to some linear function χ_S. The Corrector outputs $\chi_S(x)$ with probability $1 - 2.\varepsilon$.*

Proof: The variables x and $x+y$ are dependent but uniformly distributed. Hence

$$Prob[f(y) \neq \chi_S(y)] \leq \varepsilon$$

$$Prob[f(x+y) \neq \chi_S(x+y)] \leq \varepsilon$$

The probability of either event occurs is at most $2.\varepsilon$. If not then, $f(y).f(x+y) = \chi_S(y).\chi_S(x+y) = \chi_S(x)$, i.e. the value given by the Corrector is correct. $\square$

4 Random data

The previous complexity classes deal with the *worst-case scenario*. The constraints on the time, space, number of queries must be satisfied for all possible inputs and it is a very strong condition. A pioneer paper on the average case complexity is Levin's [13]. The SAT problem is NP-complete but seems easy on many practical inputs. The SAT conferences in recent years have shown that the problem is easy on most of the practical inputs. For example the verification that a certain property between the input x and the output y of a computer program can be reduced to very large SAT instance, which can be solved efficiently.

Machine Learning has shown that learning specific input distributions can be done efficiently, both for supervised and unsupervised learning. It is important to distinguish the structure of the data: text, trees, graphs, hypergraphs, as the concise representation of the distributions varies.

- In the case of texts, the *text mining* techniques such as Word2vec or LDA build concise representations of the distributions of texts for different languages. There are based on dimension reductions techniques for the vectors associated with each word such that $v_i.v_j$ is approximately the correlation of the words, i.e. the probability that the two words occur in the same sentence.

- In the case of graphs, *social networks* satisfy specific constraints. One of them is that the degree distribution is a power law. Which graph properties become easy on graphs with this property?

Random graphs is a classical subject with various models, such as the Erdös -Renyi model [9], where all graphs of a certain size n are uniformly distributed. The degree distribution of these graphs is however not a power law. In some other models, the Barabasi-Albert model [6], also called the *Preferential Attachment*, the degree distribution is a power law. In the *Configuration model*, we generate graphs with an aribtrary degree distribution, uniformly.

Data Science combines both algorithmic and statistical properties for various type of data. It is therefore important to understand the complexity relative to some distribution.

4.1 Efficient algorithms on specific distributions

Consider a distribution μ for social graphs. We first choose a size n with a Gaussian distribution with mean n_0 and variance 1, then choose uniformy un graph with a power law degree distribution.

Consider a graph property P on the class $\mathbf{K}$ of finite graphs and a distribution μ.

Definition 12 *A* Probabilistic Polynomial time Algorithm *A for μ, for a property $P \subseteq \mathbf{K}$ is a randomized algorithm A such that, for any structure $U \in \mathbf{K}$ as input:*
(1) If $U \models P$, then A accepts with high probability;
(2) If $U \not\models P$ when U is taken with the distribution μ, then A rejects with probability with high probability,
(3) The worst-case (average-case) time complexity is polynomial.

Notice that the probabilistic spaces are different in the conditions (1) and (2). In condition (1),

$$\mathbb{P}rob_{\Omega}[\ A \text{ accepts}] \geq 1/2 + \varepsilon$$

and Ω is the space of the random choices of A. In condition (2),

$$\mathbb{P}rob_{\mu\times\Omega}[\ A \text{ rejects}] \geq 1/2 + \varepsilon$$

and the probabilistic space is $\mu \times \Omega$.

5 Exercises

1. Show that BPP $\subseteq IP$.

2. IP protocol for $\#DNF$. Show that the language $\{(\phi, N) \ : \ \phi$ is a DNF formula with n variable and N is the number of boolean models of $\phi\}$ admits an IP protocol with n interactions. Arithmetize the formula and use the elimination of variables as in the permanent protocol.

3. IP protocol for $\#CNF$: same question as before in the case of a CNF.

4. IP protocol of graphs [8]. Consider two classical graph theoretical functions associated to a graph G_n and two distinguished nodes s and t. The function *s-t PATH* (G_n) gives the number of simple paths (without loops) between s and t. The function *s-t CONNEXITY* (G_n) gives the number of s-t connected subgraphs of G_n.

   (a) Give a recurrence relation for these two functions (isolating an arbitrary node of the graph) of the type :

   $$F(G_n) = \sum_{i=1,n^2} a_i.F(G^i_{n-1})$$

   where G^i_{n-1} is a subgraph with $n-1$ nodes. This stage corresponds to the extension stage in the protocol of the permanent.

   (b) Define a linearization stage on graphs: $G'_{n-1} = \lambda.G^1_{n-1} + (1-\lambda).G^2_{n-1}$, corresponding to the reduction stage in the protocol of the permanent.

   (c) Construct a protocol for s-t *PATH* and for s-t *CONNEXITY*.

   (d) Construct a protocol for the graph reliability function GR: given a graph G_n, two points s and t and an uncertainty function $\delta \ : \ E \to [0,1]$ which defines the probability $\delta(e)$ that an edge e exists, the function GR gives the probability that the graph is $s-t$ connected. The probabilistic space is the set of subgraphs of G_n, each with a probability which is the product of the $\delta(e)$ when e exists.

5. Linearity test of the transparent proof of 3SAT [7]. If f, g are two functions, f is δ-close to g if

   $$\mathbb{P}rob[f(x) \neq g(x)] \leq \delta$$

   (a) Show the result of Blum, Luby and Rubinfeld [7]:
   if $\delta \leq 1/3$ is a constant and if g' is a function $\{0,1\}^n \to \{0,1\}$ such that $\mathbb{P}rob_{x,y}[g'x) + g'(y) \neq g'(x+y)] \leq \delta/2$, there exists g which is δ-close of g'.

   (b) Let $a \in \{0,1\}^n$, and $p_a = \mathbb{P}rob_y[g(a) \neq g'(a+y) - g'(y)]$. Then $p_a \geq 1-\delta$.

   (c) Show that g is linear.

(d) Let $\delta < 1/3$. Show that if one of the tables A, B or D does not correspond to a linear function, then:

$$\mathbb{P}rob[\text{linearity test rejects}] \geq \delta/2$$

6. Coherence test [7].

   (a) Let $a \in \{0,1\}^n, b \in \{0,1\}^{n^2}$. Then $b = a \odot a$ if and only if for all x, x'

   $$b(x \odot x') = a \odot a(x \odot x')$$

   (b) Show that $(a \odot a)(x \odot x') = a(x).a(x') = x^T.a \odot a.x'$ where x^T is the transposed vector of x.

   (c) Show that if $a \odot a \neq b$ then $\mathbb{P}rob_{x,x'}[x^T.b.x' \neq x^T.a \odot a.x'] \geq 1/4$.

   (d) In order to compute $B(x \odot x')$, we write:

   $$B(x \odot x') = B(x \odot x' + r) - B(r) = b(x \odot x')$$

   and use the error correcting technique of [7].
   A (*self-correcting*) function for A , sc-A is defined by the algorithm: choose a random r and return $A(x+r) - A(r)$. Similarly for sc-B and sc-D. Show that if A is δ-close to a, then for all $x \in \{0,1\}^n$

   $$\mathbb{P}rob_r[\text{sc-}A(x) = a.x] \geq 1 - 2.\delta$$

   $$\mathbb{P}rob_r[A(x+r) = a.(x+r)] \geq 1 - \delta$$
   $$\mathbb{P}rob_r[A(r) = a.r] \geq 1 - \delta$$

   (e) Let $0 < \delta < 1/36$ and suppose that there exist a, b, d such as a is δ-close to A, b is δ-close to B, d is δ-close to D. Show that if $b \neq a \odot a$ and $d \neq a \odot a \odot a$ then

   $$\mathbb{P}rob[\text{Coherence test rejects}] \geq 1/4 - 9.\delta$$

7. Satisfiability test. If $0 < \delta < 1/6$ and suppose that there exist a, b, d such that a is δ-close to A, b is δ-close to B, d is δ-close to D. Show that if a is not a satisfiable valuation, then

   $$\mathbb{P}rob_r[\text{Satisfiability test rejects}] \geq 1/2 - 3.\delta$$

A Interactive protocol for the non-isomorphism of graphs

The problem of the non-isomorphism of graphs, GRAPH.NON.ISO, is defined as follows:

GRAPH.NON.ISO:
Input: two graphs (G_1, G_2) with n nodes.

Output: 1 if $G_1 \not\simeq G_2$, 0 otherwise.

If we interchange the output values, i.e. if the output is the value 0 *if* $G_1 \not\simeq G_2$, *and* 1 otherwise, we define the graph isomorphism problem, which is in the class NP. In this case, the prover transmits a permutation π which he claims satisfy $G_2 = \pi(G_1)$. The verifier checks that for all nodes a, b, we have $E_1(a, b)$ iff $E_2(\pi(a), \pi(b))$, which can be done in time $O(n^2)$.

The graph non-isomorphism is therefore in the class coNP and it seems that any prover may have to consider all the possible permutations π and make sure that none is an isomorphism: this constitutes a proof of exponential length.

We can however design an interactive protocol where the prover and the verifier exchange $O(n^2)$ bits with a probabilistic verifier bounded in time $O(n^2)$. Suppose we select a random permutation π on the finite domain D_n (the set of nodes) of the graph with a uniform distribution, with classical probabilistic techniques [14]. One can generate a graph $G' = (D_n, E')$ isomorphic to the given graph $G = (D_n, E)$ by defining $E'(i, j)$ iff $E((\pi i), \pi(j))$. This construction is used by the following interactive protocol.

Protocol for GRAPH.NON.ISO: $x = (G_1, G_2)$.

1. V picks a random $\alpha \in_r \{1, 2\}$. If $\alpha = 1$ (*resp.* $\alpha = 2$), V constructs a new graph $\overline{G}$ isomorphic to G_α by choosing a random permutation π such that $\overline{G} = \pi(G)$. The verifier V transmits this graph $\overline{G}$ to P.

2. P send $\beta \in \{1, 2\}$ to V.

3. If $\beta \neq \alpha$, then $P.V(x) = 0$, otherwise $P.V(x) = 1$.

The verifier constructs isomorphic copies $\overline{G}$ of G_1 or G_2 and asks the prover to decide if the copy $\overline{G}$ comes from G_1 or from G_2. Equivalently he asks the prover to decide the value of the random bit α which is secret. The correct answer of the prover is $\beta = 1$ if $\overline{G} \simeq G_1$ and $\beta = 2$ if $\overline{G} \simeq G_2$, which can be computed if the two graphs G_1 and G_2 are non-isomorphic. The verifier V interprets a correct answer of P as evidence that the graphs are not isomorphic, and an incorrect answer as evidence that graphs are isomorphic. Let us show that this procedure is an interactive proof.

Theorem 2 *The problem* GRAPH.NON.ISO *is in the class* IP.

Proof: If $x = (G_1, G_2) \in$ GRAPH.NON.ISO, there is a prover which never makes a mistake and can compute $\beta = \alpha$. For all random choice,

$$\mathbb{P}rob[P.V(x) = 1] = 1$$

If $x = (G_1, G_2) \notin \mathsf{GRAPH.NON.ISO}$, the graphs G_1 and G_2 are isomorphic. The new graph $\overline{G} \simeq G_1 \simeq G_2$, and the prover cannot distinguish the origin of $\overline{G}$. The probability that the prover selects $\beta \neq \alpha$ is $\frac{1}{2}$, in which case $P.V(x) = 0$ and there is no error. The probability that the prover selects $\beta = \alpha$ is also $\frac{1}{2}$, in which case $P.V(x) = 1$ and there is an error. The second condition of an IP protocol is verified. $\square$

It is clear that by repeating the test k times, the probability of error is $\frac{1}{2^k}$.

B A $\mathsf{PCP}(n^3, 1)$ protocol for 3SAT

Notice that

$$\mathcal{F} = \{f : \{0,1\}^n \to \{0,1\}\}$$

is a vector space of dimension 2^n. The set of linear functions $\mathcal{L} \subseteq \mathcal{F}$. A function $f \in \mathcal{L}$ can be written:

$$f(x) = a.x$$

where $a.x = \sum_i a_i.x_i$ (2). Observe that $\mid \mathcal{F} \mid = 2^{2^n}$ and $\mid \mathcal{L} \mid = 2^n$. One can also consider that

$$a.b = a(b) = b(a)$$

and mix functions and arguments.

If $a \neq b$, let us look at the table of $a(v)$ and the table of $b(v)$. Observe that:

$$\frac{\mid \{v : \quad a(v) = b(v)\} \mid}{2^n} = \frac{1}{2}$$

because $a.v = b.v$ iff $(a \oplus b)v = 0$, where $a \oplus b$ is the exclusive OR between vectors a and b.

Let us arithmetize the formulas on the group with two elements $GF(2)$ and not on the integers as in the protocol QBF, following the inductive definition below. Given a boolean logical formula ψ, its arithmetization is written A_ψ.

B.0.1 Boolean arithmetization

A formula can be translated into a boolean expression as follows:

- For the logical variable p_i, A_{p_i} is the variable p_i ,
- for the formula $\neg p_i$, $A_{\neg p_i}$ is $1 - p_i$,
- for the formula $(\psi_1 \wedge \psi_2)$, $A_{\psi_1 \wedge \psi_2}$ is $A_{\psi_1}.A_{\psi_2}$,
- for the formula $(\psi_1 \vee \psi_2)$, $A_{\psi_1 \vee \psi_2}$ is $A_{\psi_1} + A_{\psi_2} + A_{\psi_1}.A_{\psi_2}$.

Notice that $A_{\psi_1} + A_{\psi_2} + A_{\psi_1}.A_{\psi_2}$ corresponds also to the arithmetization of $\neg(\neg A_{\psi_1} \wedge \neg A_{\psi_2})$. The following simple lemma links the truth of a formula with its arithmetization.

Lemma 7 *For any $a \in \{0,1\}^n$, the valuation a satisfies the formula ψ iff $A_\psi(a) = 1$.*

Let us write $\psi(a) = 1$ to indicate that a is a valuation which satisfies ψ. $A_\psi(a)$ denotes the polynomial A_ψ evaluated for $p_i = a_i$. Recall that the 3SAT problem takes m clauses with three literals on $\{p_1, ..., p_n, \neg p_1, ..., \neg p_n\}$ and decides if there is a valuation.

If $C = \{C_1, ..., C_n\}$, the arithmetization is $A_C = (A_{\neg C_1}, ..., A_{\neg C_n})$, a vector of linear polynomials in $\{p_1, ..., p_n\}$ of degree 3.

Example. Consider the clauses C_1, C_2, C_3, C_4 on the propositional variables p_1, p_2, p_3, p_4.

$$\begin{array}{lcl} C_1 & : & p_1 \vee p_2 \vee p_3 \\ C_2 & : & p_2 \vee \neg p_3 \vee p_4 \\ C_3 & : & p_1 \vee p_2 \vee \neg p_4 \\ C_4 & : & \neg p_1 \vee \neg p_2 \vee \neg p_3 \end{array}$$

Let us arithmetize the negation of the clauses:

$A_{\neg C_1} : (1-p_1)(1-p_2)(1-p_3) = 1 - (p_1 + p_2 + p_3) + (p_1p_2 + p_1p_3 + p_2p_3) - p_1p_2p_3$

$A_{\neg C_2} : (1-p_2)p_3(1-p_4) = p_3 - (p_2p_3 + p_3p_4) + p_2p_3p_4$

$A_{\neg C_3} : (1-p_1)(1-p_2)p_4 = p_4 - (p_1p_4 + p_2p_4) + p_1p_2p_4$

$A_{\neg C_4} : p_1p_2p_3$

Lemma 8 *For all valuations $a \in \{0,1\}^n$, a satisfies C iff*

$$A_C(a) = (0, 0, ..., 0)$$

This lemma generalizes a similar result obtained for the arithmetization of boolean formulas in the proof of $\mathsf{IP} = \mathrm{PSPACE}$.

Corollary 1 *If C is satisfiable, there exists a valuation $a \in \{0,1\}^n$ such that*

$$\mathbb{P}rob_r[A_C.r = 0] = 1$$

If C is not satisfiable, then for any $a \in \{0,1\}^n$:

$$\mathbb{P}rob_r[A_C.r = 0] \leq 1/2$$

Example. Let $a \in \{0,1\}^n$. Consider the previous vector C and $r = (1,1,0,1)$.

$$A_C(a).r = 1 - (a_1 + a_2) + (a_1a_2 + a_1a_3 - a_3a_4) + a_2a_3a_4$$

Notice that $(a_1 + a_2) = a.(1,1,0,0)$. Write $a \odot a$ the matrix defined by $a \odot a\ (i,j) = a_i.a_j$, also considered as a vector composed of the lines of the matrix. For example, one can write: $(a_1a_2 + a_1a_3 - a_3a_4) = a \odot a.(0,1,1,0,...,0,-1,0,0,0,0)$. Similarly $a \odot a \odot a$ is a matrix of dimension 3, which can can be viewed as a vector of length n^3. There are vectors b_0, b_1, b_2, b_3, which depend on r such that:

$$A_C.r = b_0 + a.(b_1) + a \odot a.(b_2) + a \odot a \odot a.(b_3)$$

The vector b_0 is of length 1 , b_1 of length n , b_2 of length n^2 and b_3 of length n^3. A classical proof of 3SAT is a boolean model on n bits, all necessary.

Consider the function a which describes for every vector b the value $a.b$. The transparent or holographic proof is formed by the tables defining the functions a (of size 2^n), $a \odot a$ (of size 2^{n^2}) and $a \odot a \odot a$ (of size 2^{n^3}). We now describe the protocol.

The verifier computes the b_i and asks for $a.(b_1)$ which is a bit of the table a, then asks for $a \odot a.(b_2)$ which is another bit of the table $a \odot a$, and finally asks for $a \odot a \odot a.(b_3)$ which is a bit of the table $a \odot a \odot a$. The verifier asked the transparent proof π exactly three questions, checks that:

$$b_0 + a.(b_1) + a \odot a.(b_2) + a \odot a \odot a.(b_3) = 0$$

and accepts if it is true. The proof consists of the tables of the functions a , $a \odot a$ and $a \odot a \odot a$, of total size 2^{n^3}. Given three tables:

$$\begin{array}{c} A : \{0,1\}^n \to \{0,1\} \\ B : \{0,1\}^{n^2} \to \{0,1\} \\ D : \{0,1\}^{n^3} \to \{0,1\} \end{array}$$

for the functions a , $a \odot a$ and $a \odot a \odot a$, the verifier needs to test an incorrect proof. A proof can be incorrect for several reasons:

- A, B, D are not linear: a test of linearity detects it with high probability.
- A is close to a but B (resp. D) is not $a \odot a$ (resp. $a \odot a \odot a$): a test of coherence detects it with high probability.
- A is close to a , B is close to $a \odot a$ and D is close to $a \odot a \odot a$, and a is not a satisfying valuation: the previous satisfiability test detects it with high probability.

To these three cases correspond three independent tests described below. The proof that these tests satisfy the conditions of PCP is left as an exercise.

B.0.2 Linearity test

This test checks that three tables correspond to linear functions.

1. *Select random x, x' and verify that $A(x) + A(x') = A(x + x')$.*
2. *Select random y, y' and verify that $B(y) + B(y') = B(y + y')$.*
3. *Select random z, z' and verify that $D(z) + D(z') = D(z + z')$.*

Let us call sc$-A$ the (*self-correcting*) function associated with A, introduced in [7].

B.0.3 Coherence test

This test checks that the tables are coherent, i.e. $B = A \odot A$ and $D = B \odot A$.

Select random $x, x', r, r' \in \{0,1\}^n$ and verify that:

$$\text{sc-}A(x).\text{sc-}A(x') = \text{sc-}B(x \odot x')$$

$$\text{sc-}B(x).\text{sc-}A(x') = \text{sc-}D(x \odot x')$$

B.0.4 Satisfiability test

The satisfiability test considers the general condition.

1. *Select a random $r \in \{0,1\}^n$.*
2. *Compute b_i associated with r for $i = 0, 1, 2, 3$.*
3. *Verify that $b_0 + A(b_1) + B(b_2) + D(b_3) = 0$.*

We can now conclude with the **Characterization of** NP .

Theorem 3 *The problem* 3SAT *is in the class* $\mathsf{PCP}(n^3, 1)$.

Proof: If x is a set of satisfiable clauses, a prover can find a satisfying valuation a and can generate the three tables. The verifier accepts after a constant number of tests and $O(n^3)$ random bits used for the random z, z' of the linearity test.

If x is not satisfiable, the previous tests will discover an incorrect proof with high probability. The verification of the linearity and coherence tests uses technical results on self-correcting functions left as specific exercises. The verification of the satisfiability test uses Corollary 1. □

Because 3SAT is NP-complete, we conclude:

Corollary 2 *The class* NP *is contained in the class* $\mathsf{PCP}(n^3, 1)$.

In order to reduce the number of random bits from $O(n^3)$ to $O(\log n)$, we need other techniques from the theory of codes. It is however plausible that a few random bits can generate many pseudo-random bits, as random generators do.

References

[1] S. Arora, C. Lund, R. Motwani, M. Sudhan, and M. Szegedy. Proof verification and hardness of approximation problems. *IEEE Symposium on Foundations of Computer Science*, pages 14–23, 1992.

[2] S. Arora and S. Safra. Probabilistic checking of proofs. *IEEE Symposium on Foundations of Computer Science*, pages 2–13, 1992.

[3] L. Babai. Trading group theory for randomness. *Symposium on the Theory of Computing*, pages 421–429, 1985.

[4] L. Babai. Transparent proofs and limits to approximation. *Proceedings of the first European Congress of Mathematics*, pages 31–91, 1994.

[5] L. Babai, L. Fortnow, and C. Lund. Non deterministic exponential time has two-prover interactive protocols. *Computational Complexity*, 1:3–40, 1991.

[6] A. Barabasi and R.Albert. The emergence of scaling in random networks. In *Science 286*, pages 509–512, 1999.

[7] M. Blum, M. Luby, and R. Rubinfeld. Self-testing/correcting with applications to numerical problems. *ACM Symposium on Theory of Computing*, pages 73–83, 1990.

[8] J. M. Couveignes, J.F. Diaz-Frias, M. de Rougemont, and M. Santha. On the interactive complexity of graph reliability. *FSTCS International Symposium on Theoretical Computer Science, Madras*, LNCS 880:1–14, 1994.

[9] P. Erdös and A Renyi. On the evolution of random graphs. In *Publication of the mathematical institute of the Hungarian Academy of Sciences*, pages 17–61, 1960.

[10] O. Goldreich, S. Goldwasser, and D. Ron. Property testing and its connection to learning and approximation. *Journal of the ACM*, 45(4):653–750, 1998.

[11] S Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *Symposium on the Theory of Computing*, pages 291–304, 1985.

[12] S. Hougardy, H. Prömel, and A. Steger. Probabilistically checkable proofs and their consequences for approximation algorithms. Technical report, Forschunginstitut für Diskrete Mathematik, Universität Bonn, 1990.

[13] L. Levin. Average case complete problems. *SIAM Journal on Computing*, 15(1):285–286, 1986.

[14] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.

[15] C. Papadimitriou. Games against nature. *Journal of Computer and System Sciences*, 31:288–301, 1985.