Pseudorandomness and Derandomization

Evangelos Anagnostopoulos

μΠλ∀

Algorithms & Complexity II 2014

Probabilistic Algorithms

- Primality testing
- Polynomial Identity Testing

Initial conjecture: Probabilistic algorithms are more powerful than deterministic ones.

There exist problems that can be solved probabilistically in polynomial time but not deterministically.

BPP = P Conjecture

BPP has surpassed the class P as the class of problems that are considered efficiently solvable.

Two arguments to support this conjecture:

- A large number of algorithms have been implemented and work fine without access to any source of true randomness
- Every language in BPP can be non-trivially derandomized under certain assumptions

Computational Theory of Pseudorandomness

Theory introduced by Blum, Goldwasser, Micali and Yao.
Provides us with a useful conditional derandomization theorem:

“If assumption **X** is true, then every problem that can be solved by a probabilistic polynomial time algorithm can also be solved by a deterministic algorithm of running time **Y.**”

Originally, shown for
X=”there is no polynomial time algorithm for factorization”, and
Y=”time $2^{n^{\varepsilon}}$, for every $\varepsilon>0$”

Conditional Derandomization Goal

The goal became to:

- Strengthen **Y** to be polynomial time
- While the assumption **X** remains plausible

It was achieved by Impagliazzo and Wigderson in 1997.

Impagliazzo-Wigderson Result

Shown in 3 steps:

- Worst-case complexity of certain problems implies a seemingly stronger complexity of their average-case complexity (Amplification of hardness)
- Average case complexity assumption suffices to construct a certain very strong pseudorandom generator.
- This generator suffices to simulate deterministically in polynomial time every polynomial-time probabilistic algorithm.

But what is a pseudorandom generator?

Informally, it is just a map

$G:\{0,1\}^t \rightarrow \{0,1\}^m, t \ll m$, such that
if x is uniformly selected in $\{0,1\}^t$, the distribution $G(x)$
looks like the uniform distribution of $\{0,1\}^m$

Ideally, we would like $G(U_t)$ to be close to U_m in statistical distance

But this too strong of a definition... Consider the statistical test T
to be all the possible outcomes of G.

$$Pr[G(U_t)\in T]=1, \text{ but } Pr[U_m \in T]=\frac{2^t}{2^m}$$

Efficiently computable statistical tests

Computational Indistinguishability: Two distributions μ_x and μ_y over $\{0,1\}^m$ are $(K,\varepsilon)-$indistinguishable if $\forall T \subseteq \{0,1\}^m$ of circuit complexity at most K, $\left| \Pr_{x\sim\mu_x}[x \in T] - \Pr_{y\sim\mu_y}[y \in T] \right| \leq \varepsilon$

Pseudorandomness: A distribution μ_x over $\{0,1\}^m$ is $(K,\varepsilon)-$pseudorandom if it is $(K,\varepsilon)-$indistinguishable from U_m.
$\forall T \subseteq \{0,1\}^m$, of circuit complexity $\leq K$, $\left| \Pr_{x\sim\mu_x}[x \in T] - \frac{|T|}{2^m} \right| \leq \varepsilon$

Quick Pseudorandom Generator

Suppose that for every n there is a $G_n:\{0,1\}^{t(n)} \rightarrow \{0,1\}^n$ that is $(n^2, \frac{1}{n})$-pseudorandom, and that there is an algorithm G that, given n, s computes $G_n(s)$ in time $2^{O(t(n))}$.

Then G is called a $t(n)-$quick generator.

<u>logQPRG:</u> A O($\log(n)$)-quick pseudorandom generator

Application of log-QPRG

Suppose that a logQPRG exists and suppose that f is a function and A is a polynomial time probabilistic algorithm that computes f, with $Pr[A(r, I)=f(I)] \geq \frac{3}{4}$ and m = $|r|$.

Choose K to be an efficiently computable upper bound to the circuit complexity of $T=\{r: A(r, I)=f(I)\}$ and n such that $n \geq |r|, n^2 \geq K, n \geq 5$

n is polynomial in the length of I, because A runs in polynomial time.

Compute $A(G_n(s), I) \forall s \in \{0,1\}^t$ and output the most frequent value.

$Pr[A(G_n(U_t), I)=f(I)] \geq \frac{3}{4} - \frac{1}{n} > \frac{1}{2}$, because $Pr[A(U_m, I)=f(I)] \geq \frac{3}{4}$

Average case circuit complexity

A set $S \subseteq \{0,1\}^n$ is (K, ε)-hard on average if for every set T computable by a circuit of size $\leq K$ we have $Pr[1_S(x) = 1_T(X)] \leq \frac{1}{2} + \varepsilon$

A set $L \subseteq \{0,1\}^*$ is $(K(n), \varepsilon(n))$-hard on average if, for every n $L \cap \{0,1\}^n$ is $(K(n), \varepsilon(n)) -$ hard on average

Impagliazzo-Wigderson Result - Proof

Nisan and Wigderson theorem

Suppose there is a set L such that : (i) L can be decided in time $2^{O(n)}$ and (ii) there is a constant δ such that L is ($2^{\delta n}$, $2^{-2\delta n}$)-hard on average. Then a logQPRG exists.

Impagliazzo and Wigderson theorem

Suppose there is a set L such that : (i) L can be decided in time $2^{O(n)}$ and (ii) there is a constant $\delta > 0$ such that the circuit complexity of L is $\geq 2^{\delta n}$.
Then there is a set L' such that: (i) L can be decided in time $2^{O(n)}$ and (ii) there is a constant $\delta' > 0$ such that L' is ($2^{\delta' n}$, $2^{-\delta' n}$)-hard on average.

Onward to uniform hardness results

<u>Complexity class #P</u>: Counting class that outputs the number of solutions to a problem that can be solved by a NDTM with polynomial time complexity. Equivalently, outputs the number of accepting branches of such a NDTM.

<u>PERMANENT:</u> of a square matrix A nxn:

$$perm(A) = \sum_{\pi} \prod_{i=1}^{n} a_{i,\pi(i)}$$

PERNAMENT is #P-Complete

Toda's Theorem and BPP Derandomization

Toda's Theorem: $\mathrm{PH} \subseteq \mathrm{P}^{\#\mathrm{P}}$

If $\mathrm{EXP} \not\subseteq BPP$, then for every ε>0, there is a quick generator $G:\{0,1\}^{n^{\varepsilon}} \rightarrow \{0,1\}^{n}$ that is pseudorandom with respect to any P-sampleable family of n-size Boolean circuits infinitely often.

Proof: if $\mathrm{EXP} \not\subseteq \mathrm{P/poly}$, then we have proved that such a generator exists.
if $\mathrm{EXP} \subseteq \mathrm{P/poly}$, then EXP collapses to Σ_2^p and from Toda's theorem:
$\Sigma_2^p \subseteq P^{\#\mathrm{P}}$. Therefore, #P-Complete languages are complete for EXP.
PERNAMENT can be shown to be in BPP. So, BPP=EXP.

RP Derandomization

A generator H is called a hitting-set generator with respect to Any P-sampleable family of n-size Boolean circuits if, for any Probabilistic polynomial time algorithm R, where $R(1^n)$ outputs a boolean circuit of size n, there are infinitely many n s.t.

$$Pr[R(1^n) \in B_H(n)] < 1$$

If EXP $\not\subseteq$ ZPP, then for every ε>0, there is a quick hitting-set generator H: $\{0,1\}^{n^{\varepsilon}} \rightarrow \{0,1\}^n$.

If generator EASY doesn't work then ZPP=BPP

RP Derandomization

At least one of the following holds

1. RP $\subseteq$ ZPP
2. For every ε>0, every RP algorithm can be simulated in deterministic time $2^{n^{\varepsilon}}$ so that, for any polynomial time computable function f: $\{1\}^n \rightarrow \{0,1\}^n$, there are infinitely many n where this simulation is correct on the input $f(1^n)$

AM Derandomization

If E $\not\subseteq$ AM-TIME($2^{\varepsilon n}$) for some ε>0 then every language L ∈ AM has an NP-algorithm A such that for every polynomial time computable function f: $\{1\}^n \rightarrow \{0,1\}^n$, there are infinitely many n where the algorithm A correctly decides L on the input $\{1\}^n$.

This means that AM is almost as powerful as E, or AM is no more powerful than NP from the point of view of any efficient observer

Circuit lower bounds from the derandomization of MA

If NEXP $\subset$ P/poly, then NEXP = MA

EXP $\subset$ P/poly implies EXP=AM, so it sufficient to prove that NEXP $\subset$ P/poly implies NEXP=EXP.

Use generator EASY again to search for NEXP-witnesses. If the generator succeeds for every language L $\in$ NEXP then NEXP=EXP. Otherwise argue that EASY must succeed.