Derandomization

A Basic Introduction

Antonis Antonopoulos

CoReLab Seminar

National Technical University of Athens

21/3/2011

Introduction

- Randomness offered much efficiency and power as a computational resource.
- Derandomization is the "transformation" of a randomized algorithm to a deterministic one:
  Simulate a probabilistic TM by a deterministic one, with (only) polynomial loss of efficiency!
- Indications:
  1. Pseudorandomness (Randomness doesn't really exist.)
  2. "Practical" examples of Derandomization
- Possibilities concernig Randomized Languages:
  1. Randomization always help! (**BPP** = **EXP**)
  2. The extend to which Randomization helps is problem-specific.
  3. True Randomness is never needed: Simulation is possible! (**BPP** = **P**)

Facts

- Yao ,and Blum-Micali introduced the concept of hardness-randomness tradeoffs:
  If we had a hard function, we could use it to compute a string that “looks“ random to any feasible adversary (distinguisher).
  In a cryprographic context, they introduced **Pseudorandom Generators**.
- Nisam & Wigderson weakened the hardness assumption (for the purposes of Derandomization), introducing new tradeoffs between hardness and randomness.
- Impagliazzo & Wigderson proved that **P**=**BPP** if **E** requires exponential-size circuits.
- All the above results are in *non-uniform* settings, i.e. Lower Bounds of uniform classes in non-uniform models.
- Impagliazzo & Wigderson proved also a result based on Uniform complexity assumption (**BPP** $\neq$ **EXP**)!

Basic Results Outline

- **BPP** = **P**: Randomness never solves new problems (Robustness of our models).
- **BPP** = **EXP**: Randomness is powerful.

- Either:
  - **BPP** = **P**
  - No problem in **E** = **DTIME**$(2^{O(n)})$ has strictly exponential circuit complexity.

- Either:
  - **BPP** = **EXP**
  - Any problem in **BPP** has a deterministic subexponential algorithm (**SUBEXP** = $\bigcap_{\epsilon>0}$ **DTIME**(2^{n^ϵ})) that works on almost all instances.
- Simiral results for other randomized classes!

Basic Results Outline

- If we prove Lower Bounds (for some language in **EXP**), derandomization of **BPP** will follow.
- On the other hand, the existence of a quick PRG would imply a superpolynomial Circuit Lower Bound for **EXP**.
- Derandomization requires Circuit Lower Bounds:

$$\mathbf{EXP} \subseteq \mathbf{P}_{/\mathbf{poly}} \Rightarrow \mathbf{EXP} = \mathbf{MA}$$

$$\mathbf{NEXP} \subseteq \mathbf{P}_{/\mathbf{poly}} \Rightarrow \mathbf{NEXP} = \mathbf{EXP} = \mathbf{MA}$$

- It is impossible to separate **NEXP** and **MA** without proving that $\mathbf{NEXP} \not\subseteq \mathbf{P}_{/\mathbf{poly}}$.

Outline

Boolean Circuits

- A Boolean Circuit is a natural model of *nonuniform* computation.

Definition (Boolean circuits...)

For every $n \in \mathbb{N}$ an n-input, single output Boolean Circuit C is a directed acyclic graph with n sources and *one* sink.

- All nonsource vertices are called *gates* and are labeled with one of $\wedge$ (and), $\vee$ (or) or $\neg$ (not).
- The vertices labeled with $\wedge$ and $\vee$ have *fan-in* (i.e. number or incoming edges) 2.
- The vertices labeled with $\neg$ have *fan-in* 1.
- The *size* of C, denoted by $|C|$, is the number of vertices in it.

Boolean Circuits

Definition (....Boolean circuits cont.)

For every $n \in \mathbb{N}$ an n-input, single output Boolean Circuit C is a directed acyclic graph with n sources and *one* sink.

- For every vertex v of C, we assign a value as follows: for some input $x \in \{0,1\}^n$, if v is the i-th input vertex then $val(v) = x_i$, and otherwise $val(v)$ is defined recursively by applying v's logical operation on the values of the vertices connected to v.
- The *output* $C(x)$ is the value of the output vertex.
- The *depth* of C is the length of the longest directed path from an input node to the output node.

- The fixed size of the input limits our model, so we allow families of circuits to be used!

Circuit Families

Definition

Let $T : \mathbb{N} \to \mathbb{N}$ be a function. A $T(n)$*-size circuit family* is a sequence $\{C_n\}_{n\in\mathbb{N}}$ of Boolean circuits, where C_n has n inputs and a single output, and its size $|C_n| \leq T(n)$ for every n.

Definition

$\mathbf{P}_{/\mathbf{poly}}$ is the class of languages that are decidable by polynomial size circuits families. That is,

$$\mathbf{P}_{/\mathbf{poly}} = \bigcup_c \mathbf{SIZE}(n^c)$$

- $\mathbf{P} \subsetneq \mathbf{P}_{/\mathbf{poly}}$
- If $\mathbf{NP} \subseteq \mathbf{P}_{/\mathbf{poly}}$, then $\mathbf{PH} = \Sigma_2^p$ (Karp-Lipton Theorem)
- If $\mathbf{EXP} \subseteq \mathbf{P}_{/\mathbf{poly}}$, then $\mathbf{EXP} = \Sigma_2^p$ (Meyer's Theorem)

Theorem (Nonuniform Hierarchy Theorem)

For every functions $T, T' : \mathbb{N} \to \mathbb{N}$ *with* $\frac{2^n}{n} > T'(n) > 10T(n) > n$,

$$\mathbf{SIZE}(T(n)) \subsetneq \mathbf{SIZE}(T'(n))$$

Definition

For a finite Boolean Function $f : \{0,1\}^n \to \{0,1\}$, we define the (circuit) *complexity* of f as the size of the smallest Boolean Circuit computing f (that is, $C(x) = f(x), \forall x \in \{0,1\}^n$).

We can generalize the above definition for string functions:

Definition (Circuit Complexity)

For a finite Boolean Function $f : \{0,1\}^* \to \{0,1\}^*$, and $\{f_n\}$ be such that $f(x) = f_{|x|}(x)$ for every x. The (circuit) *complexity* of f is a function of n that represents the smallest Boolean Circuit computing f_n (that is, $C_{|x|}(x) = f(x), \forall x \in \{0,1\}^*$).

Circuit Families & Functions

- A super-polynomial circuit complexity for any (boolean) function in **NP**, would imply that $\mathbf{P} \neq \mathbf{NP}$.
- If f has a *uniform* (i.e. a polynomial-time algorithm that on input n produces a circuit computing f_n) sequence of polynomial-size circuits, then $f \in \mathbf{P}$.
- Also, any $f \in \mathbf{P}$ has a uniform sequence of polynomial-size circuits.
- If we prove that $\mathbf{NP} \not\subseteq \mathbf{P}_{/\mathbf{poly}}$, then we will have shown that $\mathbf{P} \neq \mathbf{NP}$
- We use this computational model, instead of TMs, because circuits are considered more direct or "pervasive".
- We also know (since 1949) that some functions require very large circuits to compute...

Existence of Hard Functions

Theorem (C.E. Shannon)

For every $n > 1$, $\exists f : \{0,1\}^n \to \{0,1\}$ that cannot be computed by a circuit C of size $\frac{2^n}{10n}$.

Proof: We use simple counting arguments:

- The number of functions $f : \{0,1\}^n \to \{0,1\}$ is 2^{2^n}
- Every circuit at size at most S can be described as a string of $9S \log S$, the nimber of circuits is at most $2^{9S \log S}$
- We set $S = \frac{2^n}{10n} \Rightarrow \cdots \Rightarrow 2^{9S \log S} \leq 2^{2^n 9n/10n} < 2^{2^n}$
- So, there exists a function that is not computed by circuits of that size!
- By more careful calculations, we can obtain a bound of: $2^n \left(1 + \frac{\log n}{n} - \mathcal{O}(1/n)\right)$ (2005).

Introduction

- Many researchers believed that circuit lower bounds are indeed the solution to the "**P** vs. **NP**".
- But the best lower bound for an **NP** language we have is $5n - o(n)$ (2005).
- Better lower bounds for some special cases:
  - **Bounded depth circuits:** $\exp\left(\Omega(n^{1/(d-1)})\right)$ (for PARITY function).
  - **Monotone circuits:** $2^{\Omega(n^{1/8})}$ (for CLIQUE), but exponential gap with general circuits.
  - **Bounded depth circuits with "counting" gates.**

Outline

Definitions

Definition (Yao-Blum-Micali Definition)

Let $G : \{0,1\}^* \to \{0,1\}^*$ be a polynomial-time computable function. Also, let $\ell : \mathbb{N} \to \mathbb{N}$ be a polynomial-time computable function such that $\forall n : \ell(n) > n$. We say that G is a *pseudorandom generator* of stretch $\ell(n)$, if $|G(x)| = \ell(|x|)$ for every $x \in \{0,1\}^*$, and for every probabilistic polynomial-time algorithm A, there exists a negligible function $\epsilon : \mathbb{N} \to [0,1]$ such that:

$$\left| Pr\left[A(G(U_n)) = 1\right] - Pr\left[A(U_{\ell(n)}) = 1\right] \right| < \epsilon(n)$$

- **Stretch Function:** $\ell : \mathbb{N} \to \mathbb{N}$
- **Computational Indistinguishability:** any algorithm A cannot decide whether a string is an output of the generator, or a truly random string.
- **Resources used:** Its own computational complexity.

Definitions

Theorem

If one-way functions exist, then for every $c \in \mathbb{N}$, there exists a pseudorandom generator with stretch $\ell(n) = n^c$.

Definition (Nisan-Wigderson Definition)

A distribution R over $\{0,1\}^m$ is an (S, ϵ)-pseudorandom (for $S \in \mathbb{N}$, $\epsilon > 0$) if for every circuit C, of size at most S:

$$|\mathbf{Pr}[C(R) = 1] - \mathbf{Pr}[C(U_m) = 1]| < \epsilon$$

where U_m denotes the *uniform distribution* over $\{0,1\}^m$
If $S : \mathbb{N} \to \mathbb{N}$, a 2^n-time computable function $G : \{0,1\}^* \to \{0,1\}^*$ is an $S(\ell)$-pseudorandom generator if $|G(z)| = S(|z|)$ for every $z \in \{0,1\}^*$ and for every $\ell \in \mathbb{N}$ the distribution $G(U_\ell)$ is $(S(\ell)^3, \frac{1}{10})$-pseudorandom.

Definitions

- The choices of the constants 3 and $\frac{1}{10}$ are arbitrary.
- The functions $S : \mathbb{N} \to \mathbb{N}$ will be considered *time-constructible* and *non-decreasing*.
- The main differences are:
  - We allow non-uniform distinguishers, instead of TMs.
  - The generator runs in exponential time instead of polynomial.

Theorem

Suppose that there exists an $S(\ell)$-pseudorandom generator for a time-constructible nondecreasing $S : \mathbb{N} \to \mathbb{N}$. Then, for every polynomial-time computable function $\ell : \mathbb{N} \to \mathbb{N}$, and for some constant c:

$$\mathbf{BPTIME}(S(\ell(n)) \subseteq \mathbf{DTIME}(2^{c\ell(n)})$$

Main Results

Theorem

- *If there exists a* $2^{\epsilon\ell}$*-pseudorandom generator for some constant* $\epsilon > 0$*, then* $\mathbf{BPP} = \mathbf{P}$.
- *If there exists a* $2^{\ell^{\epsilon}}$*-pseudorandom generator for some constant* $\epsilon > 0$*, then* $\mathbf{BPP} \subseteq \mathbf{QuasiP}$.
- *If for every* $c > 1$ *there exists an* ℓ^{c}*-pseudorandom generator, then* $\mathbf{BPP} \subseteq \mathbf{SUBEXP}$.

- We can relate the existence of PRGs with the (non-uniform) hardness of certain Boolean functions. That is, the *size* of the smallest Boolean Circuit which computes them.

Main Results

Definition (Average-case and Worst-case hardness)

For $f : \{0,1\}^n \to \{0,1\}$, and $\rho \in [0,1]$ we define the *ρ-average-case hardness* of f, denoted $H^{\rho}_{avg}(f)$, to be the largest S that for every circuit C of size at most S:

$$Pr_{x \in \{0,1\}^n}[C(x) = f(x)] < \rho$$

We define the *worst-case hardness* of f, denoted $H_{wrs}(f)$ to equal $H^{1}_{avg}(f)$, and the *average-case hardness* of f, denoted $H_{avg}(f)$ to equal: $\max\{S | H^{1/2+1/S}_{avg}(f) \geq S\}$. That is, $H_{avg}(f)$ is the largest number S such that:

$$Pr_{x \in \{0,1\}^n}[C(x) = f(x)] < \frac{1}{2} + \frac{1}{S}$$

for every Boolean Circuit C on n inputs with size at most S.

Main Results

Theorem (PRGs from average-case hardness)

Let $S : \mathbb{N} \to \mathbb{N}$ be time-constructible and non-decreasing. If there exists $f \in \mathbf{DTIME}(2^{O(n)})$ such that $\forall n : H_{avg}(f)(n) \geq S(n)$, then there exists an $S(\delta\ell)^{\delta}$-peudorandom generator for some constant $\delta > 0$.

- We can connect Average-case hardness with worst-case hardness using the following Lemma:

Theorem

Let $f \in \mathbf{E}$ be such that $H_{wrs}(f)(n) \geq S(n)$ for some time-constructible nondecreasing $S : \mathbb{N} \to \mathbb{N}$.
Then, there exists a function $g \in \mathbf{E}$ and a constant $c > 0$ such that: $H_{avg}(g)(n) \geq S(n/c)^{1/c}$ for every sufficiently large n.

Main Results

Theorem (Derandomizing under worst-case assumptions)

Let $S : \mathbb{N} \to \mathbb{N}$ be time-constructible and nondecreasing. If there exists $f \in \mathbf{DTIME}(2^{O(n)})$ such that $\forall n : H_{wrs}(f)(n) \geq S(n)$, then there exists a $S(\delta\ell)^{\delta}$-peudorandom generator for some constant $\delta > 0$.

In particular, the following hold:

1. *If there exists $f \in \mathbf{E} = \mathbf{DTIME}(2^{O(n)})$ and $\epsilon > 0$ such that $H_{wrs}(f)(n) \geq 2^{\epsilon n}$, then $\mathbf{BPP} = \mathbf{P}$.*
2. *If there exists $f \in \mathbf{E} = \mathbf{DTIME}(2^{O(n)})$ and $\epsilon > 0$ such that $H_{wrs}(f)(n) \geq 2^{n^{\epsilon}}$, then $\mathbf{BPP} \subseteq \mathbf{QuasiP}$.*
3. *If there exists $f \in \mathbf{E} = \mathbf{DTIME}(2^{O(n)})$ such that $H_{wrs}(f)(n) \geq n^{\omega(1)}$, then $\mathbf{BPP} \subseteq \mathbf{SUBEXP}$.*

Outline

Uniform Derandomization of BPP

Theorem (IW98)

If **EXP** $\neq$ **BPP**, *then, for every* $\epsilon > 0$, *every* **BPP** *algorithm can be simulated deterministically in time* $2^{n^{\epsilon}}$ *so that, for infinitely many* n*'s, this simulation is correct on at least* $1 - \frac{1}{n}$ *fraction of all inputs of size* n*.*

- That's the first (universal) Derandomization result, which implies the non-trivial derandomization of **BPP**, under a fair (but open) assumption!

But:

1. The simulation works only for infinitely many input lengths (i.o. complexity)
2. May fail on a negligible fraction of inputs even of these lengths!

Proof Outline

1. **Hard Function**: We will use a "Σ_2^p-hard" Boolean Function f with some desired properties ($\mathtt{PERMANENT}$ in our case).
2. **The Generator**: We'll construct a PRG G using the above function, similar to the NW-construction.
3. **Derandomization**: We will fix a (probabilistic) algorithm $\forall L \in \mathbf{BPP}$, and for all inputs we will run it deterministically over all outputs of G, and take the majority vote!
   If this algorithm fails to be in subexponential time, then we'll have an efficient distinguisher!
4. **Removing the Oracle:** If the above holds we have:
   - An efficient algorithm for f_n given an oracle.
   - We can "use" our construction as a **BPP** algorithm for f, by removing its oracles!

And, thus, we have a contradiction, which proves our theorem!

Uniform Derandomization of RP

Theorem (Kab01)

At least one of the following holds:

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

Uniform Derandomization of AM

Theorem (Lu00)

At least one of the following holds:

1. **AM** $=$ **NP**
2. *For any* $\epsilon > 0$*, every* **NP** *(and every co***NP***) algorithm can be simulated in deterministic time* 2^{n^ϵ} *so that, for any polynomial-time computable function* $f : \{1\}^n \to \{0,1\}^n$*, there are infinitely many n's where the simualtion is correct on the input* $f(1^n)$*.*

- Since `GNI` is in both **AM** and co**NP**, the above theorem implies that either `GNI` $\in$ **NP**, or it can be simulated in deterministic subexponential time, so that the simulation is correct with respect to any pol-time computable function $f : \{1\}^n \to \{0,1\}^n$.

Uniform Derandomization of AM

Theorem (GST03)

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

- "Gap Theorem" interpretation: Either **AM** is almost as powerful as **E**, or **AM** is no more powerful than **NP** from the point of view of any efficient observer!

Further Reading

- Sanjeev Arora and Boaz Barak, *Computational Complexity: A Modern Approach*. Cambridge University Press, 1 edition, April 2009.
- Russell Impagliazzo,*Hardness as Randomness: a survey of universal derandomization*, 2003
- Valentine Kabanets,*Derandomization: a Brief Overview*. Bulletin of the EATCS, 76:88-103, 2002.
- Russell Impagliazzo and Avi Wigderson, *Randomness vs Time: De-Randomization under a Uniform Assumption*, 1998
- Valentine Kabanets, *Easiness assumptions and hardness tests: Trading time for zero error*, 2001
- Chi-Jen Lu. *Derandomizing Arthur-Merlin Games under Uniform Assumptions*, 2000
- Dan Gutfreund, Ronen Shaltiel, and Amnon Ta-Shma. *Uniform hardness versus randomness tradeoffs for Arthur-Merlin games*, 2003

Thank You!