אוניברסיטת
אריאל
בשומרון

Computational Complexity

Lecturer: Edward A. Hirsch

https://edwardahirsch.github.io/edwardahirsch

edward.a.hirsch@gmail.com

December 24, 2025

Lecture plan

- Randomized Computation.
  - Algorithmic definitions (**ZPP**, **RP**, **BPP**).
  - Examples.
  - Probability amplification.
  - NTM-based definitions, equivalence of the definitions.
  - Discussion on probabilistic classes.
  - (To be continued in the next lectures.)

Randomized algorithms as probabilistic processes

- We consider decision problems (Yes / No).
- A randomized algorithm has access to a source of uniformly random independent bits ("random string").
- Such an algorithm can give a wrong answer with a not-so-big probability.
- Versions (to be defined later):
  - The probability of error: *mostly* interested in "*bounded*" error (small enough – to be discussed).
  - "False negative": the correct answer is Yes, but we say No.
  - "False positive": the correct answer is No, but we say Yes.
  - Both false negative and false positive answers.
  - **Zero** error (so what was the use of randomness?).

We are not talking here about approximation algorithms! The answer is exact ... but maybe wrong.

Randomized algorithms as probabilistic processes

- We consider decision problems (Yes / No).
- A randomized algorithm has access to a source of uniformly random independent bits ("random string").
- Such an algorithm can give a wrong answer with a not-so-big probability.
- Versions (to be defined later):
  - The probability of error: *mostly* interested in "*bounded*" error (small enough – to be discussed).
  - "False negative": the correct answer is Yes, but we say No. $\longrightarrow$ **RP**
  - "False positive": the correct answer is No, but we say Yes. $\longrightarrow$ **co-RP**
  - Both false negative and false positive answers. $\longrightarrow$ **BPP**
  - **Zero** error (so what was the use of randomness?). $\longrightarrow$ **ZPP**

We are not talking here about approximation algorithms! The answer is exact ... but maybe wrong.

One-sided bounded error

a probabilistic Turing machine: with a random string instead of a witness

A one-sided bounded error algorithm A for language L:

- $\forall x \in L \qquad \Pr\{A(x) = 1\} \geqslant \frac{1}{2}.$ // *False negative with probability* $\leqslant 1/2$
- $\forall x \notin L \qquad A(x) = 0.$ // *No false positives*

Class **RTime**$[t(n)]$: languages with such $O(t(n))$-time algorithms.

$$\mathbf{RP} = \bigcup_{k \in \mathbb{N}} \mathbf{RTime}[n^k]$$

One-sided bounded error: Verifying matrix multiplication

Fingerprinting

One can multiply two $n \times n$ matrices using $O(n^3)$ (obviously), or slightly faster [Strassen] or even just $O(n^{2.3715\ldots})$ arithmetic operations [Vassilevska-Williams et al, 2023]— still more than n^2.

Can one *verify* it faster?

Randomized (**co-RTime**) algorithm for verifying the product [Freivalds, 1977]:
Input: $n \times n$ matrices A, B, C.
Output: "Yes" if $A \cdot B = C$.

1. Sample a random vector $r \in \{0,1\}^n$.
2. Compute vectors $e_1 = A(Br)$ and $e_2 = Cr$. // $O(n^2)$ *operations*
3. If $e_1 = e_2$, then accept; otherwise reject.

One-sided bounded error: Verifying matrix multiplication

Fingerprinting

One can multiply two $n \times n$ matrices using $O(n^3)$ (obviously), or slightly faster [Strassen] or even just $O(n^{2.3715\ldots})$ arithmetic operations [Vassilevska-Williams et al, 2023]— still more than n^2.

Can one *verify* it faster?

Randomized (**co-RTime**) algorithm for verifying the product [Freivalds, 1977]:
Input: $n \times n$ matrices A, B, C.
Output: "Yes" if $A \cdot B = C$.

1. Sample a random vector $r \in \{0,1\}^n$.
2. Compute vectors $e_1 = A(Br)$ and $e_2 = Cr$. // $O(n^2)$ operations
3. If $e_1 = e_2$, then accept; otherwise reject.

This is a **co-RTime** algorithm. Proof:

- If $A \cdot B = C$, it accepts.
- If $A \cdot B \neq C$, let $D := AB - C$ (a matrix with nonzero entry $D_{p,s} \neq 0$). $\Pr\{\text{false positive}\}$:

$$\Pr_r\{A(Br) = Cr\} = \Pr_r\{Dr = 0\} = \Pr_r\left\{\sum_{i=1}^{n} D_{p,i} r_i = 0\right\} = \Pr_r\left\{ -\sum_{i \neq s} D_{p,i} r_i = D_{p,s} r_s \right\} \leqslant \frac{1}{2}$$

(r_i's are independent, so think like we choose r_s after $\sum_{i \neq s} D_{p,i} r_i$ is computed).

Probability amplification: one-sided version

$\frac{1}{2}$ is a huge error probability, can one do better?

[Success] probability amplification.

Let A be a **RTime** algorithm for L with error prob. $1/2$.

Amplified algorithm B.
Input: x

```
For i := 1 to k
-- if A(x) = 1 then accept.          //All attempts use independent random bits!
Reject.
```

If $x \in L$, then $\Pr\{B(x) = 0\} \leqslant \frac{1}{2^k}$.

If A runs in poly time and k is polynomial, the running time of B is still polynomial!

Probability amplification: one-sided version

$\frac{1}{2}$ is a huge error probability, can one do better?

[Success] probability amplification.

Let A be a **RTime** algorithm for L with error prob. $1/2$.

Amplified algorithm B.
Input: x

```
For i := 1 to k
-- if A(x) = 1 then accept.          //All attempts use independent random bits!
Reject.
```

If $x \in L$, then $\Pr\{B(x) = 0\} \leqslant \frac{1}{2^k}$.

If A runs in poly time and k is polynomial, the running time of B is still polynomial!

Even smaller success probability can be amplified:

In general, T attempts turn error $1 - \frac{1}{T}$ into $\left(1 - \frac{1}{T}\right)^T < \frac{1}{\mathrm{e}}$.

Thus time $\sim$ 1 / prob. of success

An important example: PIT

Problem ZeroP:

Consider an "algebraic" circuit C using arithmetic operations $(+, \cdot)$, variables x_i, integer constants. Thus it computes a polynomial in $p_C \in \mathbb{Z}[x_1, \ldots, x_n]$.

Question: Is it a **zero polynomial**? (All coefficients are zero.)

Problem PIT (Polynomial Identity Testing): easily reducible to ZeroP:

Question: Consider two circuits. Do they compute the same polynomial?

Randomized algorithm for ZeroP in **co-RP**.
Input: C.

Select random values $r_1, \ldots, r_n$ for $x_1, \ldots, x_n$.
If $C(r_1, \ldots, r_n) = 0$, then answer "Yes", otherwise answer "No".

No false negatives.

Two questions: how large r_i's and what is the probability of error?

The Schwartz–Zippel Lemma helps us

Lemma (Schwartz–Zippel)

Let $\mathbb{F}$ be a field, $0 \neq p \in \mathbb{F}[x_1, \ldots, x_n]$, $\deg p \leqslant d$. Let finite $S \subseteq \mathbb{F}$. Then

$$\Pr[p(r_1, \ldots, r_n) = 0] \leqslant \frac{d}{|S|}.$$

(This lemma is true for integers as well.)

Therefore if we choose $0 \leqslant r_i \leqslant s \cdot \deg p_C$, then the probability of error $\leqslant 1/s$.

The Schwartz–Zippel Lemma helps us

Lemma (Schwartz–Zippel)

Let $\mathbb{F}$ be a field, $0 \neq p \in \mathbb{F}[x_1, \ldots, x_n]$, $\deg p \leqslant d$. Let finite $S \subseteq \mathbb{F}$. Then

$$\Pr[p(r_1, \ldots, r_n) = 0] \leqslant \frac{d}{|S|}.$$

(This lemma is true for integers as well.)

Therefore if we choose $0 \leqslant r_i \leqslant s \cdot \deg p_C$, then the probability of error $\leqslant 1/s$.

But what is $\deg p_C$? We only know $\deg p_C \leqslant 2^{|C|}$.

Then r_i's are large, and the results of operations may reach $2^{2^{|C|}}$!

Choose $\mathbb{F} = \mathbb{F}_m$ for $m \in \mathbb{P}$, m selected at random from $2^{2|C|} \ldots 2^{4|C|}$ in reasonably many attempts (recall the Prime Number Theorem: there are $\sim N / \ln N$ primes less than N).

Two sources of errors:

- $p \neq 0$ in S.-Z. lemma, but $(r_1, \ldots, r_n)$ is a root: $\leqslant d/2^{2|C|}$, exponentially small.
- $p_C \neq 0$, but a nonzero coefficient of p_C is 0 modulo m: $\leqslant \mathrm{const} \cdot 2^{|C|} \cdot 4|C|/2^{4|C|}$, exp. small (this is because a k-bit number can have at most k prime divisors).

Two-sided bounded error

a probabilistic Turing machine: with a random string instead of a witness

A two-sided bounded error algorithm A for language L:

- $\forall x \in L \qquad \Pr\{A(x) = 1\} \geqslant \frac{3}{4}.$ // *False negative with probability* $\leqslant 1/4$
- $\forall x \notin L \qquad \Pr\{A(x) = 1\} \leqslant \frac{1}{4}.$ // *False positive with probability* $\leqslant 1/4$

Class **BPTime**$[t(n)]$: languages with such $O(t(n))$-time algorithms.

$$\mathbf{BPP} = \bigcup_{k \in \mathbb{N}} \mathbf{BPTime}[n^k]$$

Let A be a **BPTime** algorithm for L with error prob. 1/4.

Amplified algorithm B.
Input: x

`For i := 1 to k`
`-- if` $A(x) = 1$ `then` $a_i := 1$ `else` $a_i := 0$. *// All attempts use independent random bits!*
`If` $\sum_{j=1}^{k} a_i > k/2$ `then accept else reject.`

Then $\Pr\{\text{more than } k/2 \text{ errors}\} \leqslant 2^{-\Omega(k)}$. Why?

Reminder: Chernoff inequality

$$\Pr\{Z > (1+\varepsilon)pk\} < \left(\frac{e^{\varepsilon}}{(1+\varepsilon)^{1+\varepsilon}}\right)^{pk} \leq e^{-\frac{pk\varepsilon^2}{4}},$$

where $Z = \sum_{i=1}^{k} z_i$ and every z_i is an independent random variable that equals 1 with probability p and 0 with probability $1-p$.

For us z_i is an error ($a_i \neq L(x)$), in the i-th attempt, $p = \frac{1}{4}$, $\varepsilon = 1$.

Classes **RP** and **BPP** via witnesses

Bounded-error probabilistic computation

poly-bounded, poly-time verifiable

all witnesses of the same length

$L \in \mathbf{NP}$, if there is an "**NP**-relation" $R \subseteq \{0,1\}^n \times \{0,1\}^{p(n)}$ s.t. $\forall x \in \{0,1\}^*$

$$x \notin L \Rightarrow \forall w\ (x,w) \notin R,$$

$$x \in L \Rightarrow \exists w\ (x,w) \in R.$$

Thus we can think about **RP** computations in terms of computation trees.

One-sided error:

poly-bounded, poly-time verifiable

all witnesses of the same length

$L \in \mathbf{RP}$, if there is an "**NP**-relation" $R \subseteq \{0,1\}^n \times \{0,1\}^{p(n)}$ s.t. $\forall x \in \{0,1\}^*$

$$x \notin L \;\Rightarrow\; \forall w\ (x,w) \notin R,$$

$$x \in L \;\Rightarrow\; \frac{|\{w \mid (x,w) \in R\}|}{|\{\text{all } w\}|} \geqslant \frac{1}{2}.$$

Thus we can think about **RP** computations in terms of computation trees.

Classes RP and BPP via witnesses

Bounded-error probabilistic computation

One-sided error:

poly-bounded, poly-time verifiable

all witnesses of the same length

$L \in \mathbf{RP}$, if there is an "**NP**-relation" $R \subseteq \{0,1\}^n \times \{0,1\}^{p(n)}$ s.t. $\forall x \in \{0,1\}^*$

$$x \notin L \;\Rightarrow\; \forall w\ (x,w) \notin R,$$

$$x \in L \;\Rightarrow\; \frac{|\{w \mid (x,w) \in R\}|}{|\{\text{all } w\}|} \geqslant \frac{1}{2}.$$

Errorless algorithms:

$\mathbf{ZPP} = \mathbf{RP} \cap \mathbf{co\text{-}RP}$

Thus we can think about **RP** computations in terms of computation trees.

Classes RP and BPP via witnesses

Bounded-error probabilistic computation

One-sided error:

$L \in \mathbf{RP}$, if there is an "**NP**-relation" $R \subseteq \{0,1\}^n \times \{0,1\}^{p(n)}$ s.t. $\forall x \in \{0,1\}^*$

poly-bounded, poly-time verifiable

all witnesses of the same length

$$x \notin L \;\Rightarrow\; \forall w\ (x,w) \notin R,$$

$$x \in L \;\Rightarrow\; \frac{|\{w \mid (x,w) \in R\}|}{|\{\text{all } w\}|} \geqslant \frac{1}{2}.$$

any constant $0 < c < 1$

Errorless algorithms:

$\mathbf{ZPP} = \mathbf{RP} \cap \mathbf{co\text{-}RP}$

Two-sided error:

$L \in \mathbf{BPP}$, if there is an "**NP**-relation" R s.t. $\forall x \in \{0,1\}^*$

$$x \notin L \;\Rightarrow\; \frac{|\{w \mid (x,w) \in R\}|}{|\{\text{all } w\}|} \leqslant \frac{1}{4},$$

$$x \in L \;\Rightarrow\; \frac{|\{w \mid (x,w) \in R\}|}{|\{\text{all } w\}|} \geqslant \frac{3}{4}.$$

any two constants $0 < c_1 < c_2 < 1$

Thus we can think about **RP** and **BPP** computations in terms of computation trees.

Errorless randomized algorithms: Zero error (**ZPP**)

Definition 1. $\mathbf{ZPP} = \mathbf{RP} \cap \mathbf{co}\text{-}\mathbf{RP}$.
Definition 2. $L \in \mathbf{ZPP}$ if there is an errorless expected-polynomial-time algorithm for L.

Equivalence:

$\boxed{2 \Rightarrow 1}$ Let A have $\mathbb{E}\,\mathrm{time}_A(x) \leqslant p(|x|)$.

Interrupt A after $2p(|x|)$ steps and answer "no" (false negative).

$\Pr\{\mathrm{time} \geqslant 2 \cdot \mathbb{E}\,\mathrm{time}\} \leqslant \frac{1}{2}$ by Markov's inequality.

Thus we get an **RP** algorithm.

Analogously: **co**-**RP** (answer "yes" for false positives).

Errorless randomized algorithms: Zero error (**ZPP**)

Definition 1. $\mathbf{ZPP} = \mathbf{RP} \cap \mathbf{co}\text{-}\mathbf{RP}$.
Definition 2. $L \in \mathbf{ZPP}$ if there is an errorless expected-polynomial-time algorithm for L.

Equivalence:

$\boxed{1 \Rightarrow 2}$ Let B be an **RP**- and C be a **co-RP**- algorithm for L.

Run $B(x)$ and $C(x)$ until one of them produces a provably correct answer for x.

$$\mathrm{E}\, \textit{attempts} \leqslant \frac{1}{2} \cdot 1 + \frac{1}{4} \cdot 2 + \frac{1}{8} \cdot 3 + \ldots = \sum_{k=1}^{\infty} \frac{k}{2^k} = 2$$

Relation to **NP** and Closedness

Trivially $\mathbf{RP} \subseteq \mathbf{NP}$.
Thus $\mathbf{ZPP} = \mathbf{RP} \cap \mathbf{co}\text{-}\mathbf{RP} \subseteq \mathbf{NP} \cap \mathbf{co}\text{-}\mathbf{NP}$.

Where is **BPP**? We will see (it is unknown whether $\mathbf{BPP} \subseteq \mathbf{NP}$ or $\mathbf{NP} \subseteq \mathbf{BPP}$).

Closedness: Include subprocedure in the main program:

- $\mathbf{P} = \mathbf{P}^{\mathbf{P}}$.
- $\mathbf{ZPP} = \mathbf{ZPP}^{\mathbf{ZPP}}$ (use the definition with expected polynomial time) — Exercise.
- $\mathbf{BPP} = \mathbf{BPP}^{\mathbf{BPP}}$ (decrease the success probability of the oracle to exponentially small).
- $\mathbf{RP}^{\mathbf{RP}} = ?\ldots$ unknown; this way one can build a hierarchy inside **BPP**!

Errorless algorithm example: pattern matching with verification (1)

Find a pattern in a string.
Input: $p, s \in \{0,1\}^*$.
Output: "yes" if $\exists x, y$ s.t. $s = xpy$.

- **NP**: obvious.
- deterministic $|s||p|$ comparisons: also obvious.
- deterministic $O(|s|)$ time: Knuth–Morrison–Pratt algorithm.
- we will show a simple randomized $O(|s|)$-time algorithm.

Errorless algorithm example: pattern matching with verification (2)

Simple linear-time randomized "fingerprinting" algorithm:

1. Choose a random $m \in \mathbb{P}$ among at least M (to be chosen) first primes:
   due to the Prime Number Theorem there are $\sim N/\ln N$ primes less than N.
2. We identify $|p|$-bit strings with natural numbers $0\,..\,2^{|p|}-1$.
3. Compute $r := p \bmod m$.
4. For i:=1 to $|s|-|p|+1$
   4.1 Compute $r' := s[i..i+|p|-1] \bmod m$. *// Can recompute faster from the previous value*
   4.2 If $r = r'$ then
   accept.
5. Reject.

Correctness (**co-RTime**):

- If pattern p occurs in s, the algorithm accepts.
- If pattern p does not occur, there may be false positive with probability $\leqslant \frac{(|s|-|p|+1)|p|}{M}$
  (because a k-bit number has at most k prime divisors,
  so at each position i there are at most $|p|$ divisors of the number $p - s[i..i+|p|-1]$,
  $(|s|-|p|+1)|p|$ prime divisors in total).
- Choose $M \sim 2|p||s|$, thus the error probability is at most $1/2$.

Errorless algorithm example: pattern matching with verification (2)

Simple linear-time randomized "fingerprinting" algorithm:

1. Choose a random $m \in \mathbb{P}$ among at least M (to be chosen) first primes:
   due to the Prime Number Theorem there are $\sim N/\ln N$ primes less than N.
2. We identify $|p|$-bit strings with natural numbers $0\,..\,2^{|p|}-1$.
3. Compute $r := p \bmod m$.
4. For i:=1 to $|s|-|p|+1$
   4.1 Compute $r' := s[i..i+|p|-1] \bmod m$. *// Can recompute faster from the previous value*
   4.2 If $r = r'$ then if $s[i..i+|p|-1] = p$ then accept.
5. Reject.

Correctness (**co-RTime**):

- If pattern p occurs in s, the algorithm accepts.
- If pattern p does not occur, there may be false positive with probability $\leqslant \frac{(|s|-|p|+1)|p|}{M}$

ZPP modification:

$\mathbb{E}$ operations $\leqslant c|s| + O(|p|) + \frac{|s||p|}{m} \cdot |s| \cdot |p| = c|s| + O(1) + O(|p|) = c|s| + O(|p|)$
if $M \sim |s|^2|p|^2$, where c is a small constant (the number of operations to recalculate r') and $O(|p|)$ is for the initialization.