8 Randomized Complexity

18 June 2025

Prof. Dr. Sebastian Wild

CS627 (Summer 2025)
Philipps-Universität Marburg
version `2025-06-18 09:27`

Outline

8 Randomized Complexity

The Power of Randomness

We've seen examples where randomized algorithms are provably more powerful ...
but how general are such improvements?

The Power of Randomness

We've seen examples where randomized algorithms are provably more powerful . . .
but how general are such improvements?

Before we consider algorithmic design techniques, we will consider the theoretical power of randomization:

Does randomization extend the range of problems solvable by polytime algorithms?

The Power of Randomness

We've seen examples where randomized algorithms are provably more powerful . . .
but how general are such improvements?

Before we consider algorithmic design techniques, we will consider the theoretical power of randomization:

Does randomization extend the range of problems solvable by polytime algorithms?

$\rightsquigarrow$ back to *decision* problems.

8.1 Randomized Complexity Classes

Randomization for Decision Problems

- Recall: P and NP consider decision problems only

$\rightsquigarrow$ equivalently: languages $L \subseteq \Sigma^\star$

Randomization for Decision Problems

- Recall: P and NP consider decision problems only

$\rightsquigarrow$ equivalently: languages $L \subseteq \Sigma^\star$

Can make some simplifications for algorithms:

- Only 3 sensible output values: 0, 1, $\boxed{?}$
- Unless specified otherwise, allow unlimited #random bits,
  i. e., $random_A(x) = time_A(x)$ (Can't read more than one random bit per step)

Randomized Complexity Classes

Definition 8.1 (ZPP)

ZPP ***(zero-error probabilistic polytime)*** is the class of all languages L with a polytime **Las Vegas** algorithm A, i. e.,

(a) $\exists c : Time_A(n) = O(n^c)$ as $n \to \infty$ (In particular: always terminate!)

(b) $\mathbb{P}\big[A(x) = [x \in L]\big] \geq \frac{1}{2}$

(c) $A(x) \neq [x \in L]$ implies $A(x) = \boxed{?}$

Randomized Complexity Classes

Definition 8.1 (ZPP)

ZPP *(zero-error probabilistic polytime)* is the class of all languages L with a polytime **Las Vegas** algorithm A, i. e.,

(a) $\exists c \,:\, Time_A(n) \,=\, O(n^c)$ as $n \to \infty$ (In particular: always terminate!)

(b) $\mathbb{P}\big[A(x) = [x \in L]\big] \,\geq\, \frac{1}{2}$

(c) $A(x) \neq [x \in L]$ implies $A(x) = \boxed{?}$

Definition 8.2 (BPP)

BPP *(bounded-error probabilistic polytime)* is the class of languages L with a polytime **bounded-error Monte Carlo** algorithm A, i. e.,

(a) $\exists c \,:\, Time_A(n) \,=\, O(n^c)$ as $n \to \infty$

(b) $\exists \varepsilon > 0 \,:\, \mathbb{P}\big[A(x) = [x \in L]\big] \,\geq\, \frac{1}{2} + \varepsilon$

$\forall x \in \Sigma^*$

Randomized Complexity Classes

Definition 8.1 (ZPP)

ZPP *(zero-error probabilistic polytime)* is the class of all languages L with a polytime **Las Vegas** algorithm A, i. e.,

(a) $\exists c \; : \; Time_A(n) \; = \; O(n^c)$ as $n \to \infty$ (In particular: always terminate!)

(b) $\mathbb{P}\big[A(x) = [x \in L]\big] \; \geq \; \frac{1}{2}$

(c) $A(x) \neq [x \in L]$ implies $A(x) = \boxed{?}$ ◀

Definition 8.2 (BPP)

BPP *(bounded-error probabilistic polytime)* is the class of languages L with a polytime **bounded-error Monte Carlo** algorithm A, i. e.,

(a) $\exists c \; : \; Time_A(n) \; = \; O(n^c)$ as $n \to \infty$

(b) $\exists \varepsilon > 0 \; : \; \mathbb{P}\big[A(x) = [x \in L]\big] \; \geq \; \frac{1}{2} + \varepsilon$ ◀

Definition 8.3 (PP)

PP *(probabilistic polytime)* is the class of languages L with a polytime **unbounded-error Monte Carlo** algorithm: **(a)** as above **(b)** $\mathbb{P}[A(x) = [x \in L]] \; > \; \frac{1}{2}$. ◀

Error Bounds

Remark 8.4 (Success Probability)

From the point of view of complexity classes, the success probability bounds are flexible:

- BPP only requires success probability $\frac{1}{2} + \varepsilon$, but using *Majority Voting*, we can also obtain any fixed success probability $\delta \in (\frac{1}{2}, 1)$.
- Similarly for ZPP, we can use probability amplification on Las Vegas algorithms

$\rightsquigarrow$ Unless otherwise stated,

$$\boxed{\text{for BPP and ZPP algorithms } A\text{, require } \mathbb{P}\big[A(x) = [x \in L]\big] \ge \tfrac{2}{3}}$$

◀

Error Bounds

Remark 8.4 (Success Probability)

From the point of view of complexity classes, the success probability bounds are flexible:

- BPP only requires success probability $\frac{1}{2} + \varepsilon$, but using *Majority Voting*, we can also obtain any fixed success probability $\delta \in (\frac{1}{2}, 1)$.
- Similarly for ZPP, we can use probability amplification on Las Vegas algorithms

$\rightsquigarrow$ Unless otherwise stated,

$$\boxed{\text{for BPP and ZPP algorithms } A\text{, require } \mathbb{P}\big[A(x) = [x \in L]\big] \geq \tfrac{2}{3}}$$

◀

But recall: this is *not* true for **unbounded** errors and class PP.
In fact, we have the following result:

Theorem 8.5 (PP can simulate nondeterminism)

$\mathsf{NP} \cup \text{co-}\mathsf{NP} \subseteq \mathsf{PP}$.

◀

$\rightsquigarrow$ Useful algorithms must avoid unbounded errors.

PP can simulate nondeterminism [1]

Proof (Theorem 8.5):

PP always allows polytime preprocessing

Given any $L \in NP$, we can use reduction $L \leq_p SAT$ (NP-complete)

⤳ suffices to show $SAT \in PP$

(TAUT is co-NP-complete ⤳ works similarly for co-NP $\subseteq$ PP)

Given unbounded error MC algo A for SAT (polytime)

Given φ of length n over k variables

$A(\varphi)$: (1) Generate a (uniformly) random assignment $V: \{x_1, \ldots, x_k\} \to \{0,1\}$ (k random bits $O(k)$

(2) If $V(\varphi) = 1$, output 1 $O(n)$

(3) Otherwise output $B(p)$ $p = \frac{1}{2} - \frac{1}{2^{k+1}} < \frac{1}{2}$ $O(k)$

PP can simulate nondeterminism [2]

Proof (Theorem 8.5): running time polytime ✓

correctness : $\mathbb{P}[A(\varphi) = [\varphi \text{ sat.}]] \overset{?}{>} \frac{1}{2}$

- $\varphi \in SAT$ $\exists$ sat. assignment for $(x_1, \ldots, x_k)$

$\mathbb{P}[\text{ step (2) succeeds}] \geq \frac{1}{2^k}$

independence

$\mathbb{P}[A(\varphi) = 0] = \mathbb{P}[V(\varphi) = 0] \cdot \mathbb{P}[B(p) = 0]$

$\leq \left(1 - \frac{1}{2^k}\right) \cdot \left(\frac{1}{2} + \frac{1}{2^{k+1}}\right) < \frac{1}{2}$

- $\varphi \notin SAT$ $\mathbb{P}[V(\varphi) = 1] = 0$

$\mathbb{P}[A(\varphi) = 1] = 1 \cdot \mathbb{P}[B(p) = 1] = p < \frac{1}{2}$

$\Rightarrow \mathbb{P}[A(\varphi) = [\varphi \text{ sat.}]] > \frac{1}{2}$.