# PROBABILISTIC AND RANDOMIZED ALGORITHMS

STEVE ZHANG

## 1. Introduction

In the theory of computation, we often find it useful to consider non-deterministic Turing machines. Since the actual implementation of non-deterministic algorithms is impractical, they exist only in theory. However, the potential non-determinism of computers through means such as random number generators allows for the implementation of something else: *probabilistic Turing machines*. Rather than simulating every computational path of a non-deterministic Turing machine, a probabilistic Turing machine randomly selects a path based on a probability distribution at each non-deterministic choice. As a result, the probability of any computational path is the product of the probabilities of the choices made along the path.

However, the added element of chance introduces a new issue: the resulting computational path may result in a wrong answer. The measurement of the "incorrectness" of a probabilistic Turing machine is used to help classify various new complexity classes, including randomized polynomial time ($\mathsf{RP}$) and bounded-error probabilistic polynomial time ($\mathsf{BPP}$).

In this paper, we will touch on the relative positions of these new complexity classes, and go more in-depth on a special problem known as the *polynomial identity testing* problem, or PIT for short.

## 2. Probabilistic Turing Machines

**Definition 2.1.** A *probabilistic Turing machine* is the 8-tuple

$$M = (Q, \Sigma, \Gamma, \delta_1, \delta_2, q_0, q_f, q_r)$$

where

- $Q$ is the set of states,
- $\Sigma$ is the input alphabet,
- $\Gamma$ is the finite set of symbols called the tape alphabet,
- $\delta_1 : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$ is the first probabilistic transition function,
- $\delta_2 : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$ is the second probabilistic transition function,
- $q_0 \in Q$ is the initial state,
- $q_f \in Q$ is the accepting state,
- $q_r \in Q \setminus \{q_f\}$ is the rejecting state,

and at each step, the Turing machine randomly applies either $\delta_1$ or $\delta_2$ with probability $\frac{1}{2}$. This choice is made independently from all previous choices.

In essence, the computation of a probabilistic Turing machine on some input relies on flipping a coin at each transition to decide what to do. Each series of coin flips results in a unique computational branch, with the set of all computational branches forming a binary

*Date*: November 28, 2022.

tree. If every computational branch of some probabilistic Turing machine $M$ halts on some input $w$, then we can define $\Pr[M \text{ accepts } w]$ to be the fraction of computational branches which end in the accepting state. $\Pr[M \text{ rejects } w]$ is defined similarly.

The probabilistic Turing machine allows for some interesting definitions of new complexity classes. Acceptance of a string can occur with error, and the magnitude of this error can be used to distinguish between various complexity classes. Probabilistic Turing machines accepting languages in this first complexity class, $\mathsf{RP}$, are said to have "one-sided error".

## 3. RP

**Definition 3.1.** Let $t : \mathbb{N} \to \mathbb{N}$ be a function. A language $L$ is said to be in the time complexity class $\mathsf{RTime}(t(n))$ if there exists a probabilistic Turing machine $M$ that runs in $O(t(n))$, and on any input $w$,

- $\Pr[M \text{ accepts } w] \geq \frac{1}{2}$ if $w \in L$, and
- $\Pr[M \text{ accepts } w] = 0$ if $w \notin L$.

We define the complexity class randomized polynomial time ($\mathsf{RP}$) as the set of languages

$$\mathsf{RP} = \bigcup_{k=1}^{\infty} \mathsf{RTime}(n^k).$$

What is so special about the probability $\frac{1}{2}$? As it turns out, we can replace this probability with

$$1 - \frac{1}{2^{q(|w|)}}$$

for any polynomial $q$, without altering the set of languages in $\mathsf{RP}$. To show this, we define $\mathsf{RP}'$ similarly to $\mathsf{RP}$, except for all inputs $w \in L$,

$$\Pr[M \text{ accepts } w] \geq 1 - \frac{1}{2^{q(|w|)}}.$$

**Theorem 3.2.** $\mathsf{RP} = \mathsf{RP}'$.

*Proof.* The relation $\mathsf{RP}' \subseteq \mathsf{RP}$ follows immediately from the definitions, so it remains to show that $\mathsf{RP} \subseteq \mathsf{RP}'$. Let $L$ be a language in $\mathsf{RP}$, and let $M$ be its corresponding probabilistic Turing machine. The main idea is that we run $M$ on an input many times to significantly decrease our chance of error. We will create a new probabilistic Turing machine $N$ that does the following on an input $w$: run $M$ on $w$ $q(|w|)$ times, and accept if at least one of the branches accepts. Otherwise, reject. We will consider two cases: $w \in L$ and $w \notin L$.

If $w \in L$, then

$$\Pr[M \text{ rejects } w] \leq \frac{1}{2}.$$

Furthermore, $N$ rejects $w$ if and only if $M$ rejects $w$ for all $q(|w|)$ runs, so

$$\Pr[N \text{ rejects } w] \leq (\Pr[M \text{ rejects } w])^{q(|w|)} = \frac{1}{2^{q(|w|)}}.$$

Finally,

$$\Pr[N \text{ accepts } w] \geq 1 - \frac{1}{2^{q(|w|)}}.$$

If $w \notin L$, then $M$ rejects $w$ with probability 1, so $N$ will also reject $w$ with probability 1. Since $N$ runs $M$ $q(|w|)$ times, and each run takes polynomial time, $N$ must also run in polynomial time. Hence, $L \in \mathsf{RP}'$, and $\mathsf{RP} = \mathsf{RP}'$. ∎

**Theorem 3.3.** $\mathsf{P} \subseteq \mathsf{RP} \subseteq \mathsf{NP}$.

*Proof.* The first relation is obvious; a deterministic Turing machine can be interpreted as a probabilistic Turing machine where the two transition functions are identical. For the second relation, consider a probabilistic Turing machine $M$ accepting some language $L \in \mathsf{RP}$. For any input $w \in L$, $\Pr[M \text{ accepts } w] \geq \frac{1}{2}$, so there must be some accepting path. On the other hand, if $w \notin L$, then $\Pr[M \text{ accepts } w] = 0$, so there are no accepting paths. Hence, by running $M$ as a non-deterministic Turing machine and traversing all possible paths, we can deduce that $L \in \mathsf{NP}$. ■

**Definition 3.4.** We define the class $\mathsf{coRP}$ to be the $\mathsf{coRP} = \{L : \overline{L} \in \mathsf{RP}\}$. In particular, a language $L$ is in $\mathsf{coRP}$ if there exists a probabilistic Turing machine $M$ running in polynomial time such that for any input $w$,

- $\Pr[M \text{ accepts } w] = 1$ if $w \in L$, and
- $\Pr[M \text{ accepts } w] \leq \frac{1}{2}$ if $w \notin L$.

## 4. BPP

Before introducing $\mathsf{BPP}$, we will first define a term used to describe the "incorrectness" of a probabilistic Turing machine.

**Definition 4.1.** We say that a language $L$ recognized by a Turing machine $M$ with *error probability* $\epsilon$ if for all strings $w$,

- $\Pr[M \text{ accepts } w] \geq 1 - \epsilon$ if $w \in L$, and
- $\Pr[M \text{ accepts } w] \leq \epsilon$ if $w \notin L$.

In other words, the probability that $M$ is incorrect on any input is less than $\epsilon$.

**Definition 4.2.** A language $L$ is in bounded-error probabilistic polynomial time ($\mathsf{BPP}$) if there exists a probabilistic Turing machine $M$ running in polynomial time with error probability $\frac{1}{3}$. In particular,

- $\Pr[M \text{ accepts } w] \geq \frac{2}{3}$ if $w \in L$, and
- $\Pr[M \text{ accepts } w] \leq \frac{1}{3}$ if $w \notin L$.

Turing machines accepting languages in $\mathsf{BPP}$ are said to have "two-sided error". Like in the definition of $\mathsf{RP}$, the error probability $\frac{1}{3}$ is arbitrary. In fact, one can show that an error probability as high as

$$\frac{1}{2} - \frac{1}{p(|w|)}$$

for some polynomial $p$ or as low as

$$\frac{1}{2^{q(|w|)}}$$

for some polynomial $q$ will yield the same class $\mathsf{BPP}$. We will omit the proof in this paper, but it can be found in [1].

**Theorem 4.3.** *We have the following relations:*

- $\mathsf{BPP} = \mathsf{coBPP}$,
- $\mathsf{P} \subseteq \mathsf{RP} \cup \mathsf{coRP} \subseteq \mathsf{BPP}$.

*Proof.* The first relation is due to the symmetric definition of BPP. Consider a probabilistic Turing machine $M$ accepting some $L \in \mathsf{BPP}$. Then, by constructing a Turing machine $N$ that runs $M$ and reverses the output, we obtain that $\bar{L} \in \mathsf{BPP}$, so $L \in \mathsf{coBPP}$. Hence, $\mathsf{BPP} \subseteq \mathsf{coBPP}$, and similarly, $\mathsf{coBPP} \subseteq \mathsf{BPP}$, so $\mathsf{BPP} = \mathsf{coBPP}$.

The second relation is trivial by the definitions of the classes. ■

Nowadays, BPP is considered to be the most exhaustive class of problems which can be solved efficiently on modern machines, a title previously held by the complexity class P. It is still unclear whether $\mathsf{BPP} = \mathsf{P}$, and there exist languages in BPP not yet shown to be in P. However, the number of such languages is decreasing, and it is commonly believed that $\mathsf{BPP} = \mathsf{P}$. In this final section, we will take a deeper dive into one of these languages.

## 5. Polynomial Identity Testing

The polynomial identity testing problem asks when given two multivariate polynomials $p(x_1, \ldots, x_n)$ and $q(x_1, \ldots, x_n)$ over some field $\mathbb{F}$, if $p$ and $q$ are equal. This is equivalent to determining whether their difference $p - q$ is the zero polynomial. However, the problem assumes that you are only given an oracle (black box) that computes the polynomial for you. That is, you are given no information about the actual coefficients of the polynomials, but you may input values into the oracle to receive an output. As there are various models of the problem, we will establish our assumed model below.

**Question 5.1.** *(Polynomial Identity Testing Problem) Given an oracle which computes polynomial $p(x_1, \ldots, x_n)$ of degree $d$ over some field $\mathbb{F}$, does there exist $a_1, \ldots, a_n \in \mathbb{F}$ such that $p(a_1, \ldots, a_n) \neq 0$?*

We will show that this problem is in BPP, and in fact, coRP as well (under certain conditions). The heart of the proof lies in the following lemma.

**Theorem 5.2.** *(Schwartz–Zippel lemma) Let $p(x_1, \ldots, x_n)$ be a nonzero polynomial over a field $\mathbb{F}$ with degree $d$. Given some finite set $S \subseteq \mathbb{F}$, for randomly selected $a_1, \ldots, a_n \in S$,*

$$\Pr[p(a_1, \ldots, a_n) = 0] \leq \frac{d}{|S|}.$$

*Proof.* We will use induction on $n$, the number of variables in $p$. For $n = 1$, $p$ has at most $d$ roots, so the statement is trivial.

Suppose the statement holds for all $n \leq k - 1$ variables. Then, consider a polynomial $p(x_1, \ldots, x_k)$. We can rewrite $p$ as a polynomial in $x_1$ by setting

$$p(x_1, \ldots, x_k) = \sum_{i=0}^{k} x_1^i p_i(x_2, \ldots, x_k).$$

Let $m$ be the maximum value such that $p_m$ is not the zero polynomial. Since $x_1^m p_m$ has degree at most $d$, we know that $p_m$ has degree at most $d - m$. Suppose we randomly choose $a_1, a_2, \ldots, a_k \in S$. By our induction hypothesis,

$$\Pr[p_m(a_2, \ldots, a_k) = 0] \leq \frac{d - m}{|S|}.$$

Now, define the polynomial

$$q(x_1) = \sum_{i=0}^{k} x_i^i p_i(r_2, \ldots, r_k).$$

We will denote the events $p_m(a_2, \ldots, a_k) = 0$ by $A$ and $q(a_1) = 0$ by $B$. Note that if $A$ doesn't occur, then $q(x_1)$ has degree $m$, so by our induction hypothesis

$$\Pr[B \mid \neg A] \leq \frac{m}{|S|}.$$

Combining everything, we have that

$$\begin{aligned}\Pr[B] &= \Pr[B \wedge A] + \Pr[B \wedge \neg A] \\ &= \Pr[B \wedge A] + \Pr[B \mid \neg A] \cdot \Pr[\neg A] \\ &\leq \Pr[A] + \Pr[B \mid \neg A] \\ &\leq \frac{d-m}{|S|} + \frac{m}{|S|} = \frac{d}{|S|}.\end{aligned}$$

■

**Theorem 5.3.** *The polynomial identity testing problem is in* coRP *given that* $d < |\mathbb{F}|$.

*Proof.* Let $S \subseteq \mathbb{F}$ be a finite set of size $d+1$. Our algorithm will repeatedly select values $a_1, \ldots, a_n$ randomly from $S$, for a total of $d+1$ iterations. We accept if for any iteration, $p(a_1, \ldots, a_n) \neq 0$, and reject otherwise. If $p$ is indeed the zero polynomial, then our algorithm will clearly reject. Now suppose $p$ is nonzero. For each selection of values $a_1, \ldots, a_n$, we know by the Schwartz-Zippel lemma that

$$\Pr[p(a_1, \ldots, a_n) = 0] \leq \frac{d}{d+1}.$$

Hence, if we perform this selection $d+1$ times, the probability that the polynomial evaluates to 0 for all $d+1$ selections reduces to at most

$$\left(\frac{d}{d+1}\right)^{d+1} = \left(1 - \frac{1}{d+1}\right)^{d+1} \leq \frac{1}{e}.$$

Thus, when $p$ is nonzero, we accept with probability at least $1 - \frac{1}{e}$. Clearly, this satisfies the conditions of coRP, so we know that PIT is in coRP and hence BPP as well. ■

## References


[1] Friggstad, Zachary. "CMPUT 675: Complexity Theory Lecture 10 (Feb 7): Probabilistic Turing Machines" *CMPUT 675: Complexity Theory*, University of Alberta, 2019, `https://webdocs.cs.ualberta.ca/~zacharyf/courses/complexity_2019/index.html`.

[2] Harvey, Nick. "CPSC 536N: Randomized Algorithms", University of British Columbia, 2012, `https://www.cs.ubc.ca/~nickhar/W12/`.

[3] Sudan, Madhu. "6.841/18.405J Advanced Complexity Theory Lecture 6: Randomized Algorithms, Properties of BPP" *6.841/18.405J Advanced Complexity Theory*, Massachusetts Institute of Technology, 26 Feb. 2003, `https://people.csail.mit.edu/madhu/ST03/scribe/lect06.pdf`.