Randomized Algorithms

Radmehr Talebi
Khajeh Nasir Toosi University of Technology Computer Science Department
Tehran, Iran
radmehrtalebi@gmail.com

Abstract

Randomized algorithms have emerged as a powerful approach in the field of computer science, offering innovative solutions to a wide range of computational problems. By harnessing the inherent unpredictability of randomness, these algorithms provide efficient and effective methods for tackling complex computational challenges. In this paper, we delve into the realm of randomized algorithms, aiming to unravel their underlying principles and explore their applications, we shed light on the advantages and limitations of randomized algorithms and their potential impact on various domains. This research serves as a comprehensive guide for researchers and practitioners alike, providing valuable insights for leveraging the power of randomness in algorithm design and optimization.

1 Introduction

> An Algorithm most be seen to be believed
>
> *Donald Knuth*

In theory of computation one of the most crucial aspects of an algorithm is its time complexity, a good approach here is to use Randomized Algorithms. such algorithms use randomness as a tool to achieve desirable properties such as simplicity, speed, or robustness.
The design and analysis of algorithms play a pivotal role in the field of computer science, driving advancements in numerous domains and enabling efficient problem-solving. Traditional deterministic algorithms have long been the cornerstone of algorithmic research, providing precise and predictable solutions. However, the exponential growth of data and the increasing complexity of computational problems have necessitated the exploration of alternative algorithmic approaches.

Conference acronym 'XX, June 03–05, 2024, Woodstock, NY

ACM ISBN 978-1-4503-XXXX-X/18/06
https://doi.org/XXXXXXX.XXXXXXX

Randomized algorithms have emerged as a compelling solution to address the challenges posed by large-scale and computationally intensive problems. By incorporating randomness into algorithmic decision-making, randomized algorithms provide a new perspective on problem-solving, leveraging the inherent unpredictability of randomness to achieve efficient and often superior outcomes.
Furthermore, we aim to analyze the trade-offs involved in utilizing randomness, considering factors such as algorithmic efficiency, solution quality, and statistical guarantees. By comprehensively examining the power of randomness in algorithmic solutions, we aspire to foster innovation and inspire future advancements in the field of computer science. The primary objective of this paper is to delve into the realm of randomized algorithms, examining their fundamental concepts and exploring their applications. We aim to provide a comprehensive analysis of the advantages and limitations of randomized algorithms. We Introduce Monte Carlo and Las Vegas Algorithms as two main channel to Randomized Algorithms, then We expressed one-sided error and two-sided error classes to delve more into Monte Carlo Algorithms and at the end an application of Randomized Algorithms is analyzed and shown using TUTTE THEOREM.

2 NOTATIONS AND DEFINITIONS

Matching and perfect matching: Let $G = \langle V, E \rangle$ be an undirected graph. A subset $X \subseteq E$ is said to be a matching of G if no two edges in X have a common node. A matching is said to be a perfect matching if its edges cover all nodes of G.
Notation: Let permutation π^r be obtained from π by reversing all cycles.
Notation: For a perfect matching E' let $t_{E'}$ denote the product of the $a's$ corresponding to the edges of E'.

3 MONTE CARLO ALGORITHMS

A Monte Carlo algorithm is an algorithm whose output might be incorrect with a certain probability which is usually small. It is crucial to acknowledge that the precision of the Monte Carlo algorithm is contingent upon the number of samples generated. Increasing the number of samples generally enhances the accuracy of the results but also amplifies the computational time required.

3.1 ONE-SIDED VS TWO-SIDED ERROR

While the answer returned by a deterministic algorithm is always expected to be correct, this is not the case for Monte Carlo algorithms. For decision problems, these algorithms are generally classified as either false-biased or true-biased. A false-biased Monte Carlo algorithm is always correct when it returns false; a true-biased algorithm is always correct when it returns true. While this describes algorithms with one-sided errors, others might have no bias; these are said to have two-sided errors. The answer they provide (either true or false) will be incorrect, or correct, with some bounded probability.

ONE-SIDED ERROR

3.2 ONE-SIDED ERROR: Class RP

Definition: The class RP (Randomized Polynomial time) consists of all languages L that have a randomized algorithm A that runs in polynomial time such that for any input $x \in \Sigma^*$

- $x \in L \Rightarrow \Pr[A(x) \text{ accepts }] \geq \frac{1}{2}$
- $x \notin L \Rightarrow \Pr[A(x) \text{ accepts }] = 0$

3.3 ONE-SIDED ERROR: Class coRP

Definition: The class coRP consists of all languages L that have a randomized algorithm A that runs in polynomial time such that for any input $x \in \Sigma^*$

- $x \in L \Rightarrow \Pr[A(x) \text{ accepts }] = 1$
- $x \notin L \Rightarrow \Pr[A(x) \text{ accepts }] \leq \frac{1}{2}$

3.4 Two-sided error algorithms: Class PP

Definition: The class PP (Probabilistic Polynomial time) consists of all languages L that have a randomized algorithm A that runs in polynomial time such that for any input $x \in \Sigma^*$

- $x \in L \Rightarrow \Pr[A(x) \text{ accepts }] > \frac{1}{2}$
- $x \notin L \Rightarrow \Pr[A(x) \text{ accepts }] < \frac{1}{2}$
- To boost the probability of success (for example to $\frac{1}{4}$), we can repeat the algorithm and output the majority of the answers.
- Number of required repetitions could be exponential!
- Not very practical.

3.5 Two-sided error algorithms: Class BPP

Definition: The class BPP (Bounded-error Probabilistic Polynomial time) consists of all languages L that have a randomized algorithm A that runs in polynomial time such that for any input $x \in \Sigma^*$

$$\text{- } x \in L \Rightarrow \Pr[A(x) \text{ accepts }] > \frac{3}{4}$$

$$\text{- } x \notin L \Rightarrow \Pr[A(x) \text{ accepts }] < \frac{1}{4}$$

Given $x \in \Sigma^*$, the probability of error of the algorithm on input x, is at most $\frac{1}{4}$.
Fact: The error probability of the algorithm can be reduced to $\frac{1}{2^n}$ by polynomial number of repetitions.

Theorem 3.1. ZPP $\subseteq$ *RP*$\cap$ *coRP*

Proof. Let L be a decision problem in ZPP. There is a randomized algorithm A that decides $x \in L$. Let $f(n)$ be the expected running time of A on an input of size n.$f(n)$ is a polynomial function.
We want to show $L \in$ **RP** $\cap$ **coRP**. In other words:

- There is a randomized algorithm A_1 with polynomial time complexity where $x \in L \Rightarrow \Pr[A_1(x) \text{ accepts }] \geq \frac{1}{2}$ and $x \notin L \Rightarrow \Pr[A_1(x) \text{ accepts }] = 0$
- There is a randomized algorithm A_2 with polynomial time complexity where $x \in L \Rightarrow \Pr[A_2(x) \text{ accepts }] = 1$ and $x \notin L \Rightarrow \Pr[A(x) \text{ accepts }] \leq \frac{1}{2}$

Algorithm A_1 **:** Run the zero-error algorithm A on input x. Stop the execution before the running time exceeds $2f(n)$.

- If A accepts x before it is stopped, we accept x.
- If A rejects x before it is stopped, we reject x.
- Otherwise (if A is stopped before giving an answer) we accept x.

If $x \in L$, algorithm A_1 always accepts x.
If $x \notin L$, algorithm A_1 accepts x with probability at most $\frac{1}{2}$. This happens when the running time of A on x exceeds $2f(n)$. Let E be this event.
Let R be the running time of A on x. Since $E[R] \leq f(n)$
$\Pr[E] = \Pr[R \geq 2f(n)] \leq \frac{1}{2}$
(Markov inequality)
Therefore $L \in$ coRP. In the same manner, we can define the one-sided error algorithm A_2 and show that $L \in$ **RP**.
Therefore $L \in$ **RP** $\cap$ **coRP**.

□

3.6 AMPLIFICATION

For a Monte Carlo algorithm with one-sided errors, the failure probability can be reduced (and the success probability amplified) by running the algorithm k times. Consider again the Solovay–Strassen algorithm which is$\frac{1}{2}$-correct false-biased. One may run this algorithm multiple times returning a false answer if it reaches a false response within k iterations, and otherwise returning true. Thus, if the number is prime then the answer is always correct, and if the number is composite then the answer is correct with probability at least .

4 LAS VEGAS ALGORITHMS

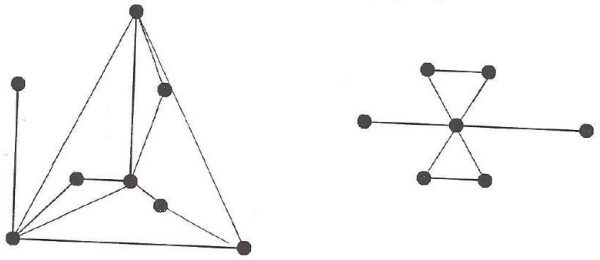
A Las Vegas algorithm is a type of randomized algorithm that always produces the correct solution, but the running time may vary depending on the input. In a Las Vegas algorithm, the randomness is used to improve the efficiency of the algorithm rather than to provide a probabilistic

guarantee of correctness. The key characteristic of a Las Vegas algorithm is that it guarantees correctness, but the running time may vary due to the random choices made during the execution. The algorithm continues to execute until it finds a solution that is guaranteed to be correct. Unlike Monte Carlo algorithms which described above whose provide a probabilistic guarantee of correctness with a fixed running time, Las Vegas algorithms focus on finding the correct solution with an efficient running time, but the actual time taken may vary depending on the input and random choices made.

5 Applications

5.1 PERFECT MATCHING ALGORITHM

We want to determine if there is a perfect matching for these graphs



There are polynomial time algorithms to decide whether a given graph has perfect matching, but none is so simple as the randomized algorithm based on so called Tutte theorem presented bellow.

5.2 Tutte matrix of a graph

Let $G = \langle V, E\rangle$ be a graph with nodes $V = \{1, 2, \ldots, n\}$, $|V|$ even. The Tutte matrix $A_G = \left\{a_{ij}\right\}_{i,j=1}^{n}$ of G is defined by

$$a_{ij} = \left\{ \begin{array}{ll} x_{ij} & \text{if } (i,j) \in E, i < j \\ -x_{ij} & \text{if } (i,j) \in E, i > j; \\ 0 & \text{if } (i,j) \notin E \end{array} \right\}$$

where x_{ij} are variables,

Theorem 5.1 (Tutte theorem:). *A graph $G = \langle V, E\rangle$, with $|V|$ even, has a perfect matching iff the determinant of the corresponding Tutte matrix is not identically zero.*

Proof. The determinant of A_G equals $\sum_\pi \sigma_\pi \prod_{i=1}^{n} a_{i\pi(i)}$ where π are permutations of $\{1, 2, \ldots, n\}$ and $\sigma_\pi = 1$ ($\sigma_\pi = -1$) if π is a product of an even (odd) number of transpositions. □

TUTTE THEOREM - OBSERVATIONS

Observation 1: For a permutation π, $\prod_{i=1}^{n} a_{i\pi(i)} \neq 0$ iff $G_\pi = \{(i, \pi(i)), 1 \leq i \leq n\}$ is a subgraph of G.

Observation 2: Permutations π with at least one odd cycle do not contribute at all to the determinant of A, because to each such permutation π there is a permutation π' such that $\prod_{i=1}^{n} a_{i\pi(i)} = -\prod_{i=1}^{n} a_{i\pi'(i)}$

Observation 3: It is sufficient to consider permutations π such that G_π consists only of even cycles.

TUTTE THEOREM - CASE ANALYSIS

Case I: $\pi = \pi^r \Rightarrow G_\pi$ consist of the cycles of length 2, π corresponds to a perfect matching E' such that $\prod_{i=1}^{n} a_{i\pi(i)} = (t_{E'})^2$.

Case II: $\pi \neq \pi^r$ In this case both π and π^r correspond to the union of two perfect matchings E, and E' obtained by alternatively selecting edges within the cycles so that

$$\prod_{i=1}^{n} a_{i\pi(i)} + \prod_{i=1}^{n} a_{i\pi^r(i)} = 2t_E t_{E'}$$

COROLLARY: $\det(A_G) = \left(t_{E'_1} + \cdots + t_{E'_k}\right)^2$ where E'_i denotes i-th perfect matching.

6 Appendix

6.1 Markov Theorem

Proposition Let X be a random variable that takes only nonnegative values. Then for any positive real number a,

$$P(X \geq a) \leq \frac{E(X)}{a}$$

provided $E(X)$ exists.

Proof. We'll prove this for discrete RVs, but the proof for continuous RVs is essentially the same, replacing sums with integrals.
By definition, $E(X) = \sum_x xP(X = x)$. We'll split this sum into two pieces, depending on whether or not $x \geq a$.

$$\begin{aligned} E(X) &= \sum_{x \geq a} xP(X = x) + \sum_{x < a} xP(X = x) \\ &\geq \sum_{x \geq a} aP(X = x) + 0 \quad (\text{ since in the first sum we assume } x \geq a) \\ &= a \sum_{x \geq a} P(X = x) \\ &= aP(X \geq a) \end{aligned}$$

□

References

[1] Michael Sipser. 1996. Introduction to the Theory of Computation. *ACM Sigact News* 27, 1 (1996), 27–29.

[2] Petar Veličković, Adrià Puigdomènech Badia, David Budden, Razvan Pascanu, Andrea Banino, Misha Dashevskiy, Raia Hadsell, and Charles Blundell. 2022. The clrs algorithmic reasoning benchmark. In *International Conference on Machine Learning*. PMLR, 22084–22102.