International Journal of Advanced Trends in Computer Applications
www.ijatca.com

A Survey on Randomized Algorithms and its Applications

Jaimeen K Patel[1], Sanjay S[2],Girish Rao Salanke N S[3]
[1,2,3] Department of Computer Science and Engineering
R V College of Engineering, Bengaluru, India-560059
[1]jaimeenpatel22@gmail.com,[2]sansa091@gmail.com,[3]girishraosalanke@gmail.com

Abstract: *Randomization has become a standard approach in algorithm design due to its efficiency and simplicity. It has been used in wide spread applications, especially in the areas of communication, cryptography, data management, and discrete optimization. This paper is aimed towards exploring the paradigms of Randomized algorithms. The paper also gives some applications of randomized algorithm in different areas and concludes giving future application areas.*

Keywords: Algorithm, randomized, deterministic**.**

I. INTRODUCTION

A randomized or probabilistic algorithm is an algorithm where the result and/or the way the result is obtained depend on chance or probability. It can also be defined as "Any algorithm that works for all practical purposes but has a theoretical chance of being wrong" [1]. These algorithms employ randomness as part of its logic.

A deterministic algorithm is one that always behaves in the same way given the same input. In other words the input completely determines the sequence of computations performed by the algorithm. On the other hand, randomized algorithms works not only on the input but also involves several random choices. The same randomized algorithm, given the same input multiple times, may perform different computations in each invocation.

The randomized algorithm can be broadly categorized as:

Monte Carlo algorithm: A randomized algorithm that may produce incorrect results. If these algorithms are run repeatedly (on the same input) with independent random choices at each time, the failure probability can be made arbitrarily small, at the expense of running time.

Las Vegas algorithm: A randomized algorithm that always produces correct results, with the only variation from one run to another being its running time.

II. DESIGN MODELS

The randomized algorithms have to work efficiently and correctly with high probability on every input, i.e., they must be reliable for each input. This requirement gives a direction towards designing randomized algorithm[2]. Below we consider two different design models of randomized algorithms [3].

2.1 Design Model I

This model considers a randomized algorithm R as a probability distribution over a finite collection of deterministic algorithms i.e.

Let I = $\{w_0, w_1 \ldots\}$ // Set of inputs
S = $\{A_0, \ldots A_n\}$ // Set of deterministic algorithms

Then R = ΔS where ΔS is the probability distribution of the set S. For any input w € I, R chooses an A_i € S at random i.e. with some probability and lets A_i work on w. After the random choice the rest of the computation is completely deterministic. This model is shown in the Figure 1.

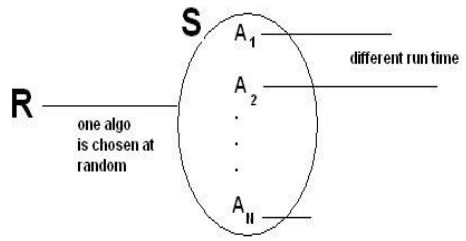

Figure 1: Model I

2.2 Design Model II

In this model we represent a randomized algorithm as a nondeterministic algorithm with a probability distribution for every nondeterministic choice i.e. repeated random choice are made after some deterministic parts of computations. This approach builds a probabilistic tree (Figure 2) where each branch can be chosen with some probability.

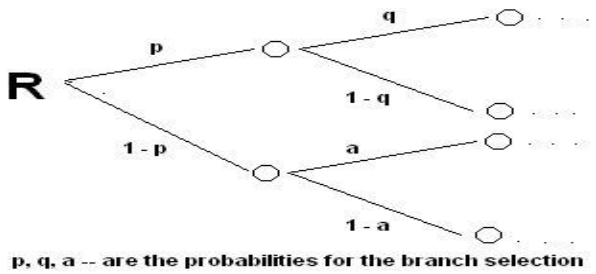

Figure 2: Model II

- Left Margin 17.8 mm (0.67")
- Right Margin 14.3 mm (0.56)
- Top Margin – 17.8 mm (0.7")
- Bottom Margin – 17.8 mm (0.7")

You should use Times Roman of size 10 for all fonts in the paper. Format the page as two columns:

- Column Width 86.8 mm (3.42")
- Column Height – 271.4 mm (10.69")
- Space/Gap between Columns - 5.0 mm (0.2").

III. Paradigms of Randomized algorithm

Here we discuss some paradigms that lie at the heart of all Randomized algorithms irrespective of the application area [4] [1].

3.1 Random Sampling

A random sample from a population representative of the population as a whole. In random sampling, also known as A random sample from a population is representative of the population as a whole. In random sampling, also known as probability sampling, every combination of items from the frame, or stratum, has a known probability of occurring, but these probabilities are not necessarily equal [4]. In Simple random sampling and Bernoulli sampling each element has an equal probability of being selected. Bernoulli sampling leads to a variable sample size, while during simple random sampling the sample size remains constant. Other examples of probability sampling include stratified sampling, multistage sampling, Poisson sampling etc.

3.2 Abundance of witness

In most cases, the algorithms are written to determine certain property on a given input. The most common example could be to find whether an input x is prime or not. To ascertain the property deterministically we will have to find witness that will practically lie in a large search space and to exhaust the space may become difficult. The alternative approach would be to construct the large space and then choose witness at random from the space.

Abundance of witness [1] has been used extensively in testing primality of a number. Fermat Little Theorem is a method used for primality testing. It says if p is a prime number and a is a positive integer less than p, then

$$a^{(p-1)} = 1 \ (\text{mod } p)$$

To check primality of p, randomly choose a, such that a < p and then calculate the above equation. If result is not 1, then p can't be prime. If the result is 1, iterate the above step and if every time we get 1 as a result, then there is a very high probability that p is prime. More the iterations we do, the higher is the probability that our result is correct. Other methods for primality test include Solovay and Strassen algorithm and Miller-Rabin primality test. Another commonly explored example in this domain is Identity Problem that checks the equivalence of two polynomials.

3.3 Foiling the adversary

Adversary is an important thing while designing algorithms. Designer design an efficient algorithm for a given problem and an adversary construct an input on which the algorithm does not work efficiently or even correctly. This is the typical way for designing and analyzing deterministic algorithms, where the adversary establishes a lower bound on the complexity of a given algorithm [4].

In case of randomized algorithm one does not know which of the possible runs of the algorithm will be chosen at random[2]. While an adversary may still be able to construct an input that is hard for one run, it is difficult to devise a single input that will defeat most of the runs of randomized algorithm. If one designs a randomized algorithm as a probability distribution over a convenient set of deterministic algorithms, then this algorithm works correctly and efficiently on every problem instance with high probability.

3.4 Fingerprinting

Fingerprinting is a method primarily for solving equivalence problems [1]. An equivalence problem can be described to find that whether two different objects representation describe the same object or not. The method uses a short representation of objects to compare the actual complex objects. The complex objects are mapped to their shorter partial representation by using techniques like hashing. These shorter versions are called as fingerprints. These fingerprints are compared to find equivalence.

This method have been successfully used in matching long strings where long strings are randomly mapped to short strings and short ones are compared. It is also been used in finding error during transfer of data in networks [5].

3.5 Random Rounding

Random rounding is a special case of random sampling. Instead of picking up samples from a set S by uniform probability distribution over S, one applies the relaxation method in order to compute another probability distribution over S, and then uses this probability distribution for random sampling in S [1]. This technique is been used in designing a randomized approximation algorithm for MAX-SAT problem.

3.6 Rapidly Mixing Markov Chains

A Markov chain is a sequence of random variables with the Markov property. Markov property says that given the present state, the future and past states are independent. Markov chains are often described by a directed graph, where the edges are labeled by the probabilities of going from one state to the other states.

A fundamental result about Markov chains says that a finite state irreducible aperiodic chain has a unique stationary distribution p and, regardless of the initial state, the time(t) distribution of the chain converges to p as t tends to infinity [6]. The counting problems such as the number of graph colorings of a given n vertex graph can be answered using the Markov chain Monte Carlo method. This method has also been used in Image analysis and matching.

3.7 Random Projections

Random projections are a powerful method for dimensionality reduction. In random projection, the original d-dimensional data is projected to a k-dimensional ($k << d$) subspace through the origin, using a random $k \times d$ matrix R whose columns have unit lengths[7]. The basis for random mapping arises from the Johnson-Lindenstrauss lemma: "If points in a vector space are projected onto a randomly selected subspace of suitably high dimension, then the distances between the points are approximately preserved". The choice of the elements of random matrix R is mostly based on Gaussian distributed but other methods are available [7]. This method have been used in applications involving data with high dimensions. Examples includes data mining, image analysis etc.

IV. APPLICATION AREAS

Randomization can be used to either find a solution to a problem or to improve a solution to a problem. Here we discuss some areas where randomization has been used.

4.1 Data Structures

Randomization can be used to improve basic data structures operations like sorting, searching or implementing dictionary operations [8]. In this context, an interesting data structures, skip lists have been proposed and used. Skip lists are an alternative to balanced binary search tress. Skip lists uses randomization in arranging items in such a way that the average search and updates operation takes O(log n) time (n been the number of items in the list).

Another commonly used example is randomized quick sort. The quick sort in its worst case takes O(n2) time and it depends on the input type. If the input is sorted then it takes this worst case scenario. So to avoid this situation we can randomize the input so that there is a high probability of not getting the sorted sequence and hence the algorithm runs in O(nlogn) running time.

4.2 Network Applications

The problem that usually arises in distributed databases is ensuring consistency of the distributed nodes [4]. Let node A and B be at different location and connected with a network. A holds a string "a" while node B holds a string "b" where a, b € {0, 1}n. The problem is to find whether a = b. The obvious way is that A sends the string "a" to B who checks whether a = b. But this takes

n bits of communication and n could be large. If we consider bandwidth to be an issue then this solution is not appropriate but this solution will always give correct result. This problem can be solved using randomization and using lesser bandwidth. The solution can be given as below [9]:

1. The strings "a" and "b" can be seen as a binary representation of the integers within the range 0, 1, 2 … 2n-1.

2. Let pi be the ith the prime number where p1 = 2, p2 = 3, p3 = 5, p4 = 7, and so on.

3. Using one of the common form of celebrated Chinese reminder theorem which says that for integers a and b and S = {1, 2, 3 …} is a finite set such that 0 <=a, b <= p(S) where

$$p(S) = \prod_{i \in S} p_i$$

then a = b iff a mod pi = b mod pi for all i

4. Node A picks up i at random from the set {1, 2, 3 … n} and calculate fa = a mod pi. Send (i, fa) to node B. The length of fa is less than length of a.

5. Node B receives the pair (i, fa) and calculates fb = b mod pi.

6. If fa = fb then a = b else a

Another problem could be controlling congestion in the networks. The routing able normally used is static in nature and the network traffic always follows a deterministic path may result in congestion. The problem can be solved using a probabilistic routing table [10] where each path has a probability associated with it.

4.3 Data Security

Data security involves a lot of Encryption algorithms which are mostly deterministic in nature. A new approach of Probabilistic Encryption [11] has been developed that works better than the deterministic algorithm. An encryption scheme normally consists of three parts: Encryption algorithm, Decryption algorithm and a key generator algorithm. In the Probabilistic Encryption approach the key generator is made probabilistic in nature. The key generator algorithm due to Goldwasser and Micali [11] is as follows:

1. Randomly select two large primes' p & q such that p

2. Let n = pq.

3. Select a pseudosquare x i.e. x is quadratic non-residue and Jacobi(x,n) = 1.

4. Public key is (n, y), the private key is (p, q).

Here the pseudosquare x required can be found by a probabilistic algorithm that picks x at random until Jacobi(x, p) = Jacobi(x, q) = -1. Further assuming the hardness of QRP, the encryption scheme has been proved to be semantically secure.

4.4 Data mining

Data mining research during the last years has led to the development of a variety of algorithms for finding frequent sequential patterns in very large databases [12]. A sequential pattern is a subsequence that appears frequently in a sequence database.

The problem of mining sequential patterns is formulated as: Assume a set L = {i1, i2 ... i_m} consisting of m distinct elements. Now consider the sequence S that is formed from elements of the set L and the number of elements in S is much larger than that in L. The problem is to find the most frequent subsequences (determined by a threshold frequency) in S. The probabilistic algorithm for mining frequent sequence as given [12] is based on the estimation of the following statistical characteristics of the sequences:

- Probability of element in the sequence
- Probability for one element to appear after another one
- Average distance between different elements of the sequence.

V. SOME MATHEMATICAL RESULTS

5.1 Probabilistic method

The probabilistic method is a combinatorial technique to use probabilistic algorithms to create objects having desirable properties and to prove that such objects exist. The technique is based on two basic observations [5]:

- If E[X] = , then there exists a value x of X, such that x >= E[X].
- If the probability of event E is larger than zero, then E exists and it is not empty.

These simple but powerful observations have been used to produce solutions to many problems. To explain the use of above let us consider the following problem:

"For any set of m clauses, there is a truth assignment of variables that satisfies at least m/2 clauses" [5].

The solution can be given as [5]: Assign every variable a random value. Clearly, a clause with k variables, has probability 1 - 2 to be satisfied. Using linearity of expectation, and the fact that even clause has at least one variable, it follows, that E[X] = m/2, where X is the random variable counting the number of clauses being satisfied. In particular, there exists an assignment for which X >= m/2.

The probabilistic method with conditional probabilities has also been used in solving the graph problems like chromatic number, crossing numbers etc.

5.2 The Chebyshev inequality

The Chebyshev inequality indicates that for a random variable X,

$$\Pr[|X - E[X]| \geq \lambda] \leq \frac{Var[X]}{\lambda^2}$$

This could be easily derived using Markov inequality. Chebyshev inequality has been useful in solving problems like finding reachability in graphs.

5.3 The Chernoff Bound

We give here a special case of Chernoff Bound: Let $X_1 \ . \ . \ . \ X_n$ be n independent random variables, such that $Pr[X_i = 1] = Pr[X_i = -1] = ½$, for i = 1, . . . , n. Let $Y = \sum X_i$ for all i from 1 to n. Then, for any $\Delta > 0$, we have

$$\mathbf{Pr}[Y \geq \Delta] \leq e^{-\Delta^2/2n}.$$

The most common application of Chernoff Bound can be seen in routing in parallel computer [5].

5.4 Lova'sz Local Lemma

Let G(V,E) be a dependency graph for events $C_1, \ . \ . \ . \ , C_n$. Suppose that there exist $x_i \in [0,1]$, for 1 <= i <= n such that:

$$\Pr[C_i] \leq x_i \prod_{(i,j)\in E} (1 - x_j)$$

then

$$\Pr\left[\cap_{i=1}^{n} \overline{C_i}\right] \geq \prod_{i=1}^{n} (1 - x_i) .$$

The lemma has been used to find solutions to many complex problems. K-SAT is an example of a problem for which solution has been proposed using this lemma [11].

VI. TESTING RANDOMIZED ALGORITHM

Randomized algorithms are sometimes much better in terms of performance or sometimes even complexity from their deterministic alternatives. Formal verification methods [14] are normally used to verify randomized algorithms as black-box testing is limited to statistical error reports. Also we need to look into new sets of errors that arise due the implementation difficulty of randomized algorithm. In case of Randomized algorithms one has to be avoid errors mathematically and also have to remember that there is a certain uncertainty associated while using randomized algorithm.

Future Directions

In the recent past new randomized approximation algorithms have been developed for the problems in graphs, Boolean algebra, convex bodies and factoring integers [1]. Below we give are some problem areas where randomization can help:

- Distributed graph algorithms are mostly deterministic which in turn are exponentially slower. The use of randomization in these algorithms can lead to simpler and faster algorithms. Problems like message routing, Byzantine agreement etc can be solved using randomization [13].
- The selection and sorting algorithms have been using randomization for achieving better run time results. The problem is to improve randomized algorithms so as to have lower bounds on the number of comparisons. Another direction is to de-randomize these algorithms i.e. to find deterministic algorithms with the same running time bounds.

- There are many open problems in the geometrical computation. For example, given a simple polygon P and a triangle T , is there a sub-quadratic algorithm that determines whether T can be placed inside P , allowing both translations and rotations. Another related problem is: Given a segment e and a polygon P with holes, determine, in sub-quadratic time, whether e can be placed inside P, using translations and rotations.

Simulation and gaming involves a lot of randomness that can be readily applied in the areas like component based software engineering, agent based software engineering and other emerging trends in our line of business. Testing is another area where the use of randomness can be used to forecast potential problems.

REFERENCES

[1].Karpinski, M., and Verbeek, R., On Randomized versus Deterministic Computation, Proc. ICALP '93, LNCS 700 (1993), Springer-Verlag, pp. 227–240.
[2].Hromkovich, "Design and Analysis of Randomized Algorithms: Introduction to Design Paradigms", Springer, Berlin–Heidelberg (2005).
[3]. J. Hromkovic , I. Z´amecnikov," Design and Analysis of Randomized Algorithms: Introduction to Design Paradigms", Texts in Theoretical Computer Science, an EATCS series. Springer, first edition, July 2005. ISBN: 978-3-54023-949-9.
[4].R. Motwani and P. Raghavan, " Randomized Algorithms", Cambridge University Press, 1995.
[5]. Sariel Har-Peled,"Lectures notes on Randomized Algorithms" ,2005.
[6].D. Randall. Rapidly mixing markov chains with applicationsin computer science and physics. Computing in Science and Engineering, 8(2):30–41, 2006.
[7].E. Bingham, H. Mannila, "Random Projection in Dimensionality Reduction: Applications to Image and Text Data", Proc. ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining, pp. 245-250, 2001.
[8].Michael T. Goodrich and Roberto Tamassia ,"Using randomization in the teaching of data structures and algorithms", The proceedings of the thirteenth SIGCSE technical symposium on Computer science education, Pages 53-57 ,1998
[9].Mihir Bellare, "Notes on Randomized Algorithms", Computer Science and Engineering, UCSD
[10].E. Bingham, H. Mannila, "Random Projection in Dimensionality Reduction: Applications to Image and Text Data", Proc. ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining, pp. 245-250, 2001.
[11].Fuchsbauer, G.J, "An Introduction to Probabilistic Encryption", Osjecki Matematicki List, 2006. 6 p. 37-44.
[12].Romanas Tumasonis and Gintautas Dzemyda. Analysis of the statistical characteristics in mining of frequent sequences. In Intelligent Information Systems, pages 377–386, 2005.
[13].Rajiv Gupta, Scott A. Smolka, Shaji Bhaskar, "On Randomization in Sequential and Distributed Algorithms",Journal ACM Computing Surveys (CSUR) ,Volume 26 Issue 1, March 1994 ,Pages 7-86 .
[14].Hurd J, "Formal Verification of Probabilistic Algorithms", PhD thesis, University of Cambridge (2002).