International Institute of
Information Technology, Hyderabad

Term Project

Techniques for Generating Random Numbers in Parallel

Final Report

Author: Kalp Shah

April 30, 2021

Contents

Chapter 1

Introduction

Randomization is defined to be the study of making something random, which can be mathematically defined as :

Definition 0.1

For a function $\mathcal{G} : \mathcal{D} \rightarrow \mathcal{R}$, if $\mathcal{G}$ is random, then :

$$P(\mathcal{G}(x) = y) = \frac{1}{\|\mathcal{R}\|}$$

The definition above says that the probability function for any input x to give an output y is uniformly random.

For a sequence of random numbers generated, two properties must hold :

- The values are uniformly distributed over a defined interval
- It is impossible to predict future values based on the past or present ones

Note 0.1

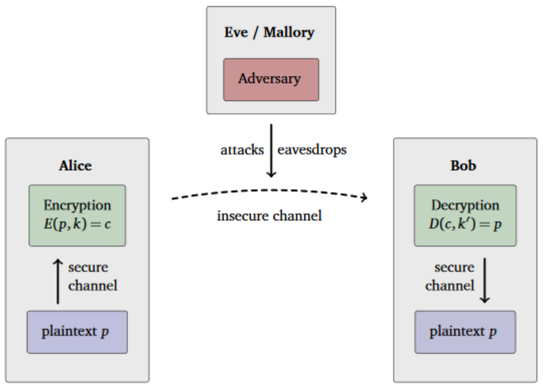
All the definitions provided here is for a function to give a random output, which is what the report will be focussing on.

1.1 Importance of Randomness

1.1.1 Cryptography

One of the most important use of random numbers is in cryptography. Cryptography is a study of techniques for secure communication in presence of third party adversary. Formulation and explaination of a cipher can easilty explain the requirement of random numbers.

A general communication channel looks as follows:



In this it can be seen that c, which is the ciphertext contains more information (higher entropy) than m but has no intrinsic meaning in itself. Therefore, it can be anything.

$E(m,k)$ is often randomized, in the case that even if the system is faulty and gets the same m and k, the ciphertext c will not be the same, thus protecting the information.

Thus, $E(m)$ is a random number generator, which is why random number generators are important in the field of cryptography.

1.1.2 Monte Carlo

Another important application of random number generation is in Monte Carlo simulation methods. Monte Carlo methods are set of algorithms that repeatedly use random sampling to obtain deterministic results.

One of the simplest and most used example of a Monte Carlo simulation problem is the determination of the value of π. The code to calculate is as below:

```
# Imports
import numpy as np

# Total number of samples to be taken
N = 100000
pCiricle = 0
pt_x = 0
pt_y = 0

for i in range(N):
        # Random values of X & Y Taken
        # X,Y in [0,1]
        pt_x = np.random.uniform()
        pt_y = np.random.uniform()

        if np.square(pt_x) + np.square(pt_y) <= 1:
            # Point in circle if x^2 + y^2 < 1
            pCiricle += 1

    # Area of Circle in First Quadrant
    # is pi/4 (r = 1), hence pi = 4*density of
    # points in circle
    pi = 4 * (pCiricle/N)
    print(pi)
```

From this, it can be seen that a deterministic value (π) is being found using random sampling of x and y points.

1.2 Pseudorandomness

Any algorithm that exists is a set of defined steps that will always give a deterministic output. Therefore, generation of true random numbers is impossible for a normal computer algorithm.

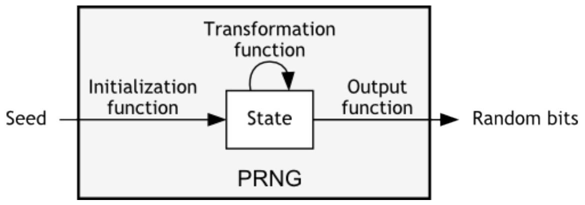
Rather if for the required amount of length, the sequence appears to be stastically random, then the job is done. An algorthm that provides with that kind of sequences are known as psuedo random number generators.

An example of one such function is :

$$\mathcal{G}(x) = g^x \mod m$$

In this, the parameters are g and m. This is a psuedo random number generator in the sense that if a predictive (non random) sequence of numbers are input as x, then the output sequence will appear to be random.

The general structure of a psuedo random number generator is as follows:



Thus, the sequence generated by a computer algorithm is not truly random, but for a particular length of output appears to be random, known as a psuedo random sequence.

Chapter 2

Sequential RNG

In this section, the most common sequential RNGs are introduced. This is required as RNGs that work in parallel are extensions of SRNGs, and understanding a simpler system before introducing a N processor generality wil be the logical and simpler way to understand PRNGs.

2.1 Linear Congruential Generators

Linear Congruential Generators, abbreviated as LCGs are defined as follows:

Definition 1.1

For a sequence of numbers $X_1, X_2, \ldots, X_n$ with n^{th} number being X_n, the sequence is :

$$X_n = aX_{n-1} + c \mod m$$

Where a, c, m are parameters.

This is one of the most common RNG, which has been used for a long time. One advantage that this construct has is that it is very quick in generation when m is chosen to be a power of 2.

The disadvantage of the system are that it produces highly correlated lower order bits for intervals that are power of 2. It also has the issue that the modulus cannot be greater than the precision fo the machine.

One modification that is commonly used for the algorithm is generalization of LCG, known as Multiple Recursive Generators (MRGs), which are defined as:

Definition 1.2

For a sequence of numbers $X_1, X_2, \dots, X_n$ with n^{th} number being X_n, the sequence is :

$$X_n = \sum_{i=1}^{n-k} a_i X_i + c \mod m$$

Where a_i, c, m, k are parameters.

This is a recursion based some of the previous values rather than just the last one. The benefit of this construct is that the randomness of the system increases, lesser the k is, on the downside that it takes a longer time to compute.

2.2 Lagged-Fibonacci Generators

Lagged-Fibonacci Generators, abbreviated as LFGs are defined as follows:

Definition 2.1

For a sequence of numbers $X_1, X_2, \dots, X_n$ with n^{th} number being X_n, the sequence is :

$$X_n = X_{n-p} \odot X_{n-q} \mod m$$

Where p, q, m are parameters and $\odot$ is any binary operation (Add, Multiply ,XOR).

From the formulation itself it can be seen that the usage of the construct requires $p(p > q)$ initial values to be already present.

The binary operation can be selected by the user and it has been proven that multiply operation provides the most amount of randomness followed by the addition operation and lastly followed by XOR operation.

The major benifit of the construct is that it is extremely quick in computing the value. Its period can also be made arbitrarily wide by just changing the value of p.

2.3 Combined Generators

This is a system which just combines two psuedo random sequences, which in turn gives a random sequence which would have a better quality of randomness. Hence, the formulation is as follows :

Definition 3.1

For two sequences X_n and Y_n, making a combined sequence S_n as follows:

$$S_n = X_n + Y_n \mod m$$

Where m is the parameter.

2.4 Conclusion

It is visible that psuedo random techniques used are not that complicated, but rather tricky. The combination of the use of recursion and modulo provides us with psuedo random numbers.

The improvement in quality can be performed by performing binary operations on these sequences, which if having different parameters would have different patterns thus obscuring the pattern even more, making the operated sequence even more random.

Chapter 3

Parallel RNG

Parallel algorithms for RNGs is just a generalization or an extension of the sequential RNGs. Two main techniques are used for such generalization, first is splitting and the other being parametrization.