Monte Carlo Markov Chain

for Solving a Roadblock Problem

Ashraf Abd El Raouf
Dr. at *Misr International University*, *Computer Science*
Cairo, Egypt
Ashraf.raouf@miuegypt.edu.eg

Hagar Sobeah and Mahmoud Hazem
Asst. at *Misr International University*, *Computer Science* Cairo, Egypt
Hager.sobeah@miuegypt.edu.eg , Mahmoud.ezzat@miuegypt.edu.eg

AbdulRahman Sherif, Khaled Yehia, Moshir Ashraf, Sara Yasser
Computer Science, *Misr International University*
Cairo, Egypt
AbdulRahman2100536@miuegypt.edu.eg, Khaled2103399@miuegypt.edu.eg, Moshir2107190@miuegypt.edu.eg, Sara2100416@miuegypt.edu.eg

***Abstract*—We researched and implemented a highly efficient program based on a Randomized Monte Carlo Markov Chain algorithm using C++ to predict the probable location of a police roadblock after a set amount of hours given it's initial set-up location. Randomized approach is an approach that solves problems by using methods that rely on a certain degree of randomness to find solutions with lower time complexity than other traditional approaches. Monte Carlo algorithms are a type of randomized algorithms that try to reach a sufficient solution within a specified number of attempts. Police roadblocks usually cause massive traffic jams that most people would rather avoid but have no possible guaranteed method to evade them despite their best efforts. Markov Chains are stochastic models which are used to predict the next possible transition based on the current state of affairs. A typical MCMC algorithm uses the Metropolis method to generate the probability that a select state would be the current state after a number of iterations.**

***Index Terms*—Algorithms, MCMC, Randomized, Approaches, Markov Chain , C++**

I. Introduction

Algorithms are just detailed descriptions of the sequential steps that must be taken to produce a given result. They are among the most often used tools for knowledge exchange. Multiple issues that people used to face are now swiftly resolved and disentangled through the amalgamation of different algorithms. Throughout the paper, a roadblock problem is explored and simplified using a randomizing approach that tries to expect the probable state, or intersection, of the registered police truck. After many trials and errors, multiple debates and queries and a long list of complaints, its been decided to tackle the issue using a Metropolitan Monte Carlo Markov Chain algorithm. Perhaps, however, the most obvious method to implement would be to use :

```
srand(time(0)) % state_number;
```

. Nonetheless, after multiple accesses, the usage of non-deterministic data-types has been opted instead; to hasten the execution of the program. By adapting a randomized **MCMC** algorithm, multiple issues can now be more easily maneuvered.

In the first section, the Randomized approach is going to be reviewed and its pros and cons are briefly going to be mentioned, while comparing its efficiency to that of the Brute Force. After that, in the second section, the Monte Carlo Algorithm, an example of a randomized algorithm, will be covered along with assessing the distinctions between that algorithm and another randomized algorithm : *Las Vegas*. In the succeeding section, the paper tries to simplify the algorithmic issue by tackling a roadblock problem; an uncomplicated illustration for the problem. From there on,the fifth section explains the math behind the Markov Chain; by underlining its formulas and equations, alongside mentioning various practices of and for the model. Ultimately, in the final section, the implementation of the algorithm as a C++ code is demonstrated by analyzing the main functions and different data-types used.

II. Randomized Approach

A randomized algorithm is an algorithm that employs a degree of randomness as part of its logic or procedure. The algorithm typically uses uniformly random bits as an auxiliary input to guide its behavior, in the hope of achieving good performance in the "average case" over all the possible random choices determined by the random bits; thus either

the running time, or the output (or both) are random variables. "The last decade has witnessed a tremendous growth in the area of randomized algorithms. During this period, randomized algorithms went from being a tool in computational number theory to finding widespread application in many types of algorithms". [4] In simpler words, the design of these algorithms, merely depends on a number of non-deterministic random choices, which can vary based on the inputs entered.

Generally speaking, there are two widely known advantages of said approach; how fast and swift are the performances of the algorithms implemented, compared to their counterparts and how simpler and less complex it is to execute. In fact, the only apprehensively acclaimed drawback of the approach is that in some implementations, the outcome displayed to the user might not be the globally optimum, or accurate, solution. However, it is important to state that these sub solutions are still, in their own right, good solutions.

TABLE I
COMPARISON BETWEEN BRUTE FORCE & RANDOMIZED APPROACHES

Approach	**Brute Force**	**Randomized**
Usage & Properties	When there is no other algorithm available to speed up the process	When presented with a time or memory constraint, and an average case solution is an acceptable output
	Tries all possible solutions to a problem	Uses randomness and probabilistic expectations
	Deterministic	Non-deterministic
Time Complexity	O(n^2)	O($\sqrt{n}$)

III. MONTE CARLO ALGORITHM

Monte Carlo algorithms are a class of computational algorithms that rely on repeated random sampling to obtain numerical results. The underlying concept is to use randomness to solve problems that might be deterministic in principle. However, these algorithms are often resource-restricted, meaning that they require a limited amount of computational resources. Albeit ,they are most used in situations where the accuracy of the solution is not critical, or where the cost of obtaining a more accurate solution is too high; provided that these solutions outputted to the user may not be the most optimum ones available, within a certain margin of error nonetheless. Particularly, the more samples that are used, the more accurate the Monte Carlo algorithm will be. Although this may be true, using more samples can also make the algorithm more computationally expensive.

TABLE II
COMPARISON BETWEEN MONTE CARLO & LAS VEGAS

Algorithm	**Monte Carlo**	**Las Vegas**
Usage & Properties	Might not always give a correct solution but an average solution is always presented as a faster result	Always gives a correct and accurate solution but usually takes a long time while computing it
	Used when it is important to get an answer quickly	Used when it is important to get a precise answer
Run-time	Fixed	Not fixed
Time Complexity (Quick Sort)	O(mn)	O(n^2)

"A Monte Carlo simulation is a technique that handles non-normal distributions, complex algorithms and correlations between input factors for the model in question. In this case, a distribution is determined for each parameter (see below). Then data are generated for each distribution, and these data are used as input for the model to produce output, these two steps being repeated as many times as is reasonably necessary to achieve an outcome curve or distribution in its own right." [2]

Among the many methods of the Monte Carlo, perhaps the easiest in implementation and the one with the highest efficiency, is the Metropolis method which generates a sequence of random samples from a target distribution. Then, at each step, the algorithm proposes a new sample from a proposal distribution that was the easiest to sample from.
By using the algorithm to simulate the behavior of a system, it became possible to manipulate the stock market for instance, as it can be used to predict the future price of a stock or to calculate the risk of an investment.

IV. ROADBLOCK PROBLEM

A common daily problem facing commuters is traffic jams that are caused by police roadblocks. As per protocol, these roadblocks are only allowed to change their location at regular time intervals throughout the day. The trucks are often set up at intersections to intercept unsettling or shady looking vehicles. While these roadblocks facilitate law enforcement elements' jobs, they present a major inconvenience for suburbanites travelling through the city who wish to maneuver their neighborhoods without hassle and with haste. with this in mind, many disgruntled travellers have tried to find effective workarounds; such as timing their trips out of peak hours, as well as using public transportation. Be that as it may, all their attempts were to no avail as there was no consistent reassurance provided to avoid the roadblocked and congested intersections. Among all possible algorithms and models that have been utilized to attempt to

predict probable areas of congestion only one out-shined the others, the MCMC model.

V. Markov Chains

The concept of Markov chains was first brought to light by the renowned Russian mathematician Andrey Andreyevich Markov in 1906. First things first, Markov chains are Stochastic models which means they describe a mathematical object defining a random sequence of variables whose index is based on an interpretation of time. They presume a chain of probable future state transitions in the next iteration wholly based on the current state. The probability distribution of a state to state transition is usually described as an N*N transition matrix. Furthermore, in said transition matrix , the value of index[i][j] displays the likely-hood of transitioning from state i to state j. Markov chain models describe the transition of a certain object from one state to another based on preassigned probability percentages dependent on the current state. To put it simply, "What happens next depends only on the state of affairs now".

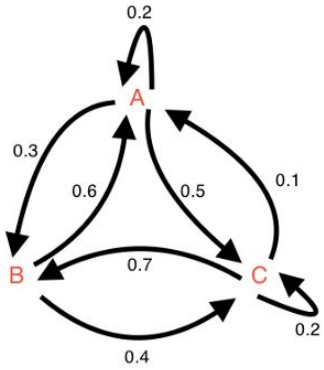

Fig. 1. Marcov Chain Representation

A. Formulas & Equations [3]

- **Definition:** The process $X_{(t)} = X_0, X_1, X_2, ...$ is a discrete- time Markov chain if it satisfies the Markov property:

$$(X_{n+1} = s|X_0 = x_0, X_1 = x_1, ..., X_n = x_n) = P(X_{n-1} = s|X_n = x_n) \quad (1)$$

- The state space of this Markov chain is S = {0,1,...,n}, so the transition matrix will look like this when

$$\text{P} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0.6 & 0 & 0.4 & 0 & 0 & 0 \\ 0 & 0.6 & 0 & 0.4 & 0 & 0 \\ 0 & 0 & 0.6 & 0 & 0.4 & 0 \\ 0 & 0 & 0 & 0.6 & 0 & 0.4 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \text{ n = 5 \& p = 0.4}$$

B. Applications

1) Real Life Implementations:

- Natural Language Processing MCMC techniques play a role in tasks like language modeling and speech recognition. These algorithms help generate coherent and contextually appropriate sentences by iteratively sampling words or phrases based on the probabilities learned from training data.

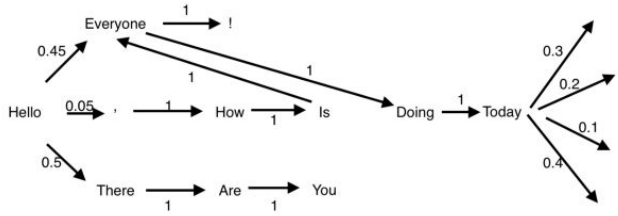

Fig. 2. Natural Language Model

- Code Breaking A state prison officer reached out to the statistics department at Stanford University for assistance in decoding messages used by inmates. These messages held a set of unfamiliar symbols with hidden meanings. The statisticians at Stanford came to notice that each symbol represented an English letter. To unravel the code, they put to use the MCMC algorithm, which involved iteratively analyzing the symbols and their potential letter correspondences. After numerous iterations, they successfully deciphered the code and understood the messages. [1]

2) Movie Adaptations:

- Finding Nemo In the movie, Marlin and Dory use a search algorithm to locate Nemo in the vast ocean. The algorithm simulates various possible locations based on available information and adjusts its predictions iteratively. This iterative process resembles the concept of MCMC, where the algorithm explores different states and updates its estimates based on the observed data.
- Sherlock Holmes Sherlock Holmes, the famous detective, solves complex crimes by inferring the truth from available evidence. He employs a process of elimination, refining his hypotheses based on new information, and gradually converges on the correct solution. This iterative refinement process aligns with the underlying principles

of MCMC algorithms, where probabilistic inference and iterative sampling help approximate the true state from observed data.

VI. Implementation and Tests

A. Data Types

1) Rand: A function of the math.h, cmath, and cstdlib libraries that generates random numbers. However upon further research, other data types emerged that provide a higher degree of randomness with less processing cost and faster results.

2) MT19937: The MT19937 , developed by Akuji Nishimura and Makoto Matsumoto in 1997, is a data-type in the "std namespace" that uses Mersenne Twister pseudo-random number generators to generate random values. Mersenne Twisters,named after the French friar Marin Mersenne, use a seed of 19,937 bit long to generate 32 bit values. Mersenne Twisters have a period of $2^{1}9937$ - 1 which is a Mersenne prime (a prime number defined as $M_n = 2^n$ - 1 and have great time complexity as they are based on binary operations. Moreover, they use a scarce amount of the memory space, owing to the fact that it only consumes 624 words of the working area. They primarily work by initializing a state based on the given seed value, then pulling a random single integer , shifting some bits and finally generating a number.However, a state has a finite set of state integers (624) after which the algorithm reloads by twisting the current integers and generating a new state containing another 624 new state numbers. To keep it short, the mt19937 is a pseudo-random number generator with an internal state of 19937 bits,624 integers, that generates a multitude of random numbers efficiently while avoiding repetitions.

3) Random Device: random device is a uniformly distributed integer random number generator used for producing non-deterministic random numbers. It can be used as a seed for the mt19937 data-type to increase the randomness of the generated outcomes.

B. Random Walk

```
void randomwalk()
{
    counter.assign(msize * msize, 0);
    int randomState = initialState;
    cout << "Start at " << randomState << endl
        ;
    random_device rd;
    mt19937 gen(rd());
    uniform_int_distribution<> rowDist(0,
        msize - 1);
    uniform_int_distribution<> colDist(0,
        msize - 1);
    int randomi, randomj, temp;

    for (unsigned short i = 0; i < hours; i++)
         {
        cout << "\nHour :  " << i + 1;
        do {
            temp = randomState;
            randomi = rowDist(gen);
            randomj = colDist(gen);
        } while (transitionMatrix[temp][
            randomi][randomj] == 0);
        randomState = convert(randomi, randomj
            );
        cout << "\nRandom state : " <<
            randomState << " with (" <<
            randomi << ", " << randomj << ")"
            << endl;
        counter[randomState]++;
    }
    cout << endl << endl;
}
```

To begin with , we initialize a 2-dimensional vector to count the number of times a certain node has been selected. Then , we create a random device using it as a seed for the MT 19937. Following up, we then use uniform distribution to match the range of our matrix after which the program enters a loop that runs for a number of iterations equal to the number of hours specified by the user. In each iteration of the loop, the algorithm tries to generate the next possible state to transition to while adhering to the predefined probability matrix. After randomly choosing the next state to transition to the next state is set as the current state , and a new random next state is generated until the number of iterations reaches the number of hours. Finally, when function execution completes successfully we are left with the number of times a certain node has been visited which is then used to calculate the probability that a node is the current state at a given time.

VII. Conclusion

Generally speaking, as has been stated above, the Markov Chain case is still to this present day, one of the most troubling yet important problems; there will never be a hundred percent guarantee of a one way solution or path to the problem. Although this may be true, the **MCMC** model has been a game-changing sky-rocketing method that facilitate the resolving of the probabilistic chain. For one thing, the randomized algorithm has had a major role in the training and programming of multiple complex *AI* architectures. To put it differently, the paper's roadblock problem is merely a simple application and implementation of the Monte Carlo Markov Chain algorithm, which as explained above, the papers would strongly recommend to use and apply its rules whenever faced with a problem of such sorts.

References

[1] Departments of Mathematics and Statistics, *Persi Diaconis*, "The Markov Chain Monte Carlo Revolution" , 2000.

[2] The National Institute of Standards and Technology, *Sean Salleh*, "The National Institute of Standards and Technology", August, 2013.

[3] New York University, *Holmes-Cerfon, Miranda*, "Lecture 2: Markov Chains", 2019.

[4] Cambridge University Press, *Motwani ,Raghavan*,Randomized algorithms,2013