eXclusive e-JOURNAL
ISSN 1339-4509

ECONOMY & SOCIETY & ENVIRONMENT

Decision-making and problem solving in real life using parallel Monte Carlo Simulations.

Mgr. Miroslav Karahuta

University of Presov in Presov
Faculty of Management
Department of Quantitative Methods And Managerial Informatics
Slovenská 67
080 01 Prešov
Slovakia
karahuta@gmail.com

Abstract
Every day, millions of people solve problems and make decisions based on their intuition or other soft skills. Using some of the decision-making methods, we can solve many of those situations more precisely. One of such a method can be Monte Carlo Simulations. The first part of this article is short introduction to Monte Carlo with few examples. The second part contains definition of real-life personal logistic problem and solution with Monte Carlo Simulation. Algorithm is implemented using MPI and C++ and simulated on computer cluster. The speedup of parallel computation is compared with serial algorithm.

Key words Monte Carlo simulations, Monte Carlo Method, problem solving, logistic

1. INTRODUCTION

Monte Carlo methods are computational algorithms that rely on repeated random sampling to obtain numerical results. For example, by running simulations many times over in order to calculate those same probabilities heuristically just like actually playing and recording the results in a real casino situation. Monte Carlo methods are mainly used in three distinct problems: optimization, numerical integration and generation of samples from a probability distribution.

There is no consensus on how Monte Carlo should be defined. For example, Ripley [7] defines most probabilistic modeling as stochastic simulation, with Monte Carlo being reserved for Monte Carlo integration and Monte Carlo statistical tests. Sawilowsky [8, pp. 218-225] distinguishes between a simulation, a Monte Carlo method, and a Monte Carlo simulation. Kalos and Whitlock [4] point out that such distinctions are not always easy to maintain. For example, the emission of radiation from atoms is a natural stochastic process. It can be simulated directly, or its average behavior can be described by stochastic equations that can themselves be solved using Monte Carlo methods.

2. SIMPLE EXAMPLES

2.1 Birthday paradox

Simple examples of Monte Carlo Simulations are really very simple. We can use one of the most classic problems in probability – birthday paradox. This is simple problem with quite long answer.

The basic idea is to find out, if in a group of *n* people at least two people share the same birthday. There are three assumptions:

a) We assume that all 365 days of the year (for simplicity, we ignore leap years) are equally likely birthdays.
b) There are no twins in the room.
c) There are no more than 365 people in the room. In case of $n > 365$, the probability is equal to 1.

2.1.1 Classical approach

We can ask every person in the group of *n* people to reveal their birthday, we get an ordered n-tuple of birthdays. The sample space **Ω** is the set of all n-tuples of birthdays and $|\Omega| = 365^n$.
Based on assumptions, each outcome of Ω is equally likely to occur with probability $1/365^n$. We would like to know probability of event **A**={*there is at leas one match among the n birthdays*}. Since there are a lot of cases to consider while computing |A| directly, it is simpler to compute complementary event **A'**={*there is no match among the n birthdays*}.
The outcomes in **A'** are ordered arrangements of n numbers chosen from 365 numbers without repetitions. Therefore:

$$|A'| = 365 \times 364 \times 363 \ldots (365 - n + 1)$$

The probability of no match is:

$$P(A') = \frac{365 \times 364 \times 363 \ldots (365 - n + 1)}{365^n}$$

And the probability of at least one match is:

$$P(A) = 1 - \frac{365 \times 364 \times 363 \ldots (365 - n + 1)}{365^n}$$

Even for small groups (small *n*) the probability is surprisingly large:

- For n=23, P(A) = 0,507. For group of 23 people, there is more than half chance of having the same birthday.
- For group of 106 people (n=106), the probability of birthday match is almost certain: P(A) = 0,9999999.

2.1.2 Monte Carlo solution

Using Monte Carlo simulation to solve this problem can be summarized into simple 4-step algorithm [5, p. 2]:

1) Pick *n* random numbers (days) in the range: 1 - 365.
2) Check if any of the *n* numbers are equal.
3) Go back to step 1 and repeat 10 000 times.
4) Report the fraction of trials with sucesfull match.

2.2 Estimating π with Buffon's needle experiment

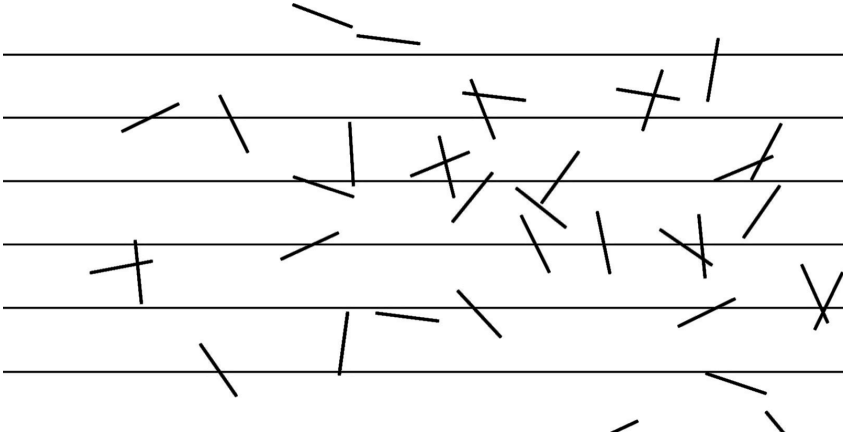
Buffon's Needle is one of the oldest problems in the field of geometrical probability. The problem was first posed by the French naturalist Buffon in 1733 [2, pp. 43-45] and reproduced with solution by Buffon in 1777 [1, pp. 100-104].

Buffon's Needle refers to simple Monte Carlo method for the estimation of the value of π. The idea assumes there is a paper with parallel lines that are equally spaced (e.g. 10 cm) and needle, which is also 10 cm long. During the experiment, the needle is dropped on the paper, which produce two possibilities:

a) The needle touches or crosses one of the lines.
b) The needle does not cross or touch the lines.

Experiment is based on numerous repetition of the needle dropping (see. Fig. 1) and keeping track of both the total number of times the needle is randomly dropped on the paper *(N*) and the number of times it crosses (or touches) a line (*C*). If experiment is repeated enough, the number *2N/C* approaches the value of π.

Fig. 1: Buffon's Needle experiment.



Source: Author's design.

3. PARALLEL COMPUTING - TECHNICAL AND SOFTWARE SOLUTION

When designing the implementation of parallel computing algorithm, it is necessary to, in addition to the algorithm itself, choose the right platform that enables the optimal use of allocated resources and thus accelerate the calculation. There are several possibilities: from the large, millions dollars, "supercomputers" to simply connect standard PCs using a standard network and the necessary software. The new and increasingly popular option is renting capacity in a Cloud. This method allows us dynamically allocate technical resources and optimize expenses.
During the development stage, the PC with dual-core processor was used. This solution allowed us to use parallel computation and communication with 2 physical units (cores) without multicore simulations. The algorithm was tested on a cluster located within the United Institute of Nuclear Research in Dubna in Russia.

As important as parallel platform are appropriate software tools. To implement the algorithm, we chose a higher programming language C++. Greater abstraction language allowed us to focus more on the algorithm itself, while a high-performance of language meet our requirements. The implementation of the algorithm uses only the standard C++ libraries with the exception of MPI (Message Passing Interface) and Mersenne Twister pseudo-random number generator implementation in C++ by Jasper Bedaux.

Because of the required summarization of calculated data from different cores into the main one, it was necessary to choose efficient interprocess communication mechanism. Unofficial standard in this area - MPI - was selected. Standard specification defines the MPI library subroutines, which contain communication functions to transfer data between processors, functions performing collective operations over some set of processors, and many other functions that deal with the transmission of messages and dynamically creating new processes. There are three popular implementation of MPI. There is a highly portable implementation of the MPI standard - MPICH, for a variety of parallel and distributed computing environments. OpenMP model is suitable for multiprocessor systems with shared memory.
For this implementation we chose OpenMPI library, for its high adaptation on high-performance clusters in the world, including the one in Dubna. The MPI core functions have been used for interprocess communication including MPI Reduce.

3.1 Mersenne Twister

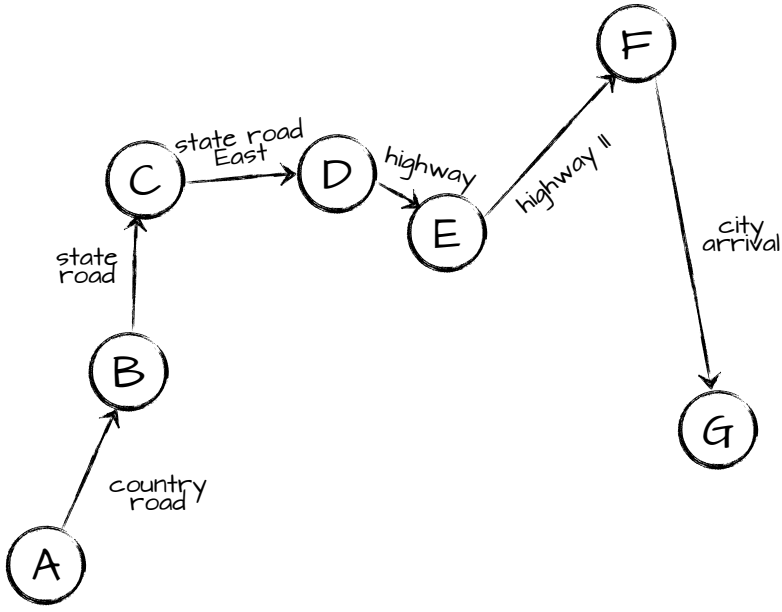
Monte Carlo Simulations are based on probability and randomness and thus algorithm requires high quality number generator. Mersenne Twister (MT) is a pseudorandom number generating algorithm developed by Makoto Matsumoto and Takuji Nishimura (alphabetical order) in 1996/1997. [6, pp. 3-30]
Main advantages of MT and reasons for using in Monte Carlo simulations are:

- Fast generation.
- It is designed with consideration on the flaws of various existing generators.
- Efficient use of the memory.
- Far longer period and far higher order of equidistribution than any other implemented generators. It is proved that the period is 2^{19937}-1, and 623-dimensional equidistribution property is assured.
- High quality implementation in C++ by Jasper Bedaux.

4. MORNING MEETING PROBLEM

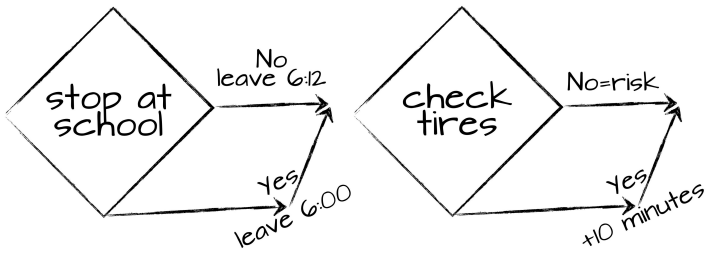
There are much more complex problems which can be solved by Monte Carlo Simulations. But for demonstration purposes, the "morning meeting problem" was chosen. Its simplicity allowed us to demonstrate algorithms and ideas behind Monte Carlo Simulation. “The morning meeting problem” is based on ideas in Armando Jeronymo’s article “Running Monte Carlo Simulations in PHP” [3].
Problem is based on logistic route planning from home to the meeting in the other city. Person, call him David, has a very important Monday morning meeting. David must not be late, so he is planning his route and timing in advance. He split the path into several parts, based on type of road (see Fig. 2).
After that, he realized that there might be some extra events before or during the ride. These events might affect arrival time and thus make him come to meeting late. In this model, we choose two events. First one is taking child to school and second one is about checking tires.
First event is purely based on personal feelings. If he takes daughter to school, he will have to leave 10 minutes earlier but it will help his wife. Checking tires is different situations. If he will stop and check the tires, he would be certain of their pressure. Otherwise, he would have to travel with the uncertainty. Poor tire pressure could have an effect on driving stability and speed. Both events are sketched on Fig. 3.

Fig. 2: Sketch of stages of route.



Source: Author's design.

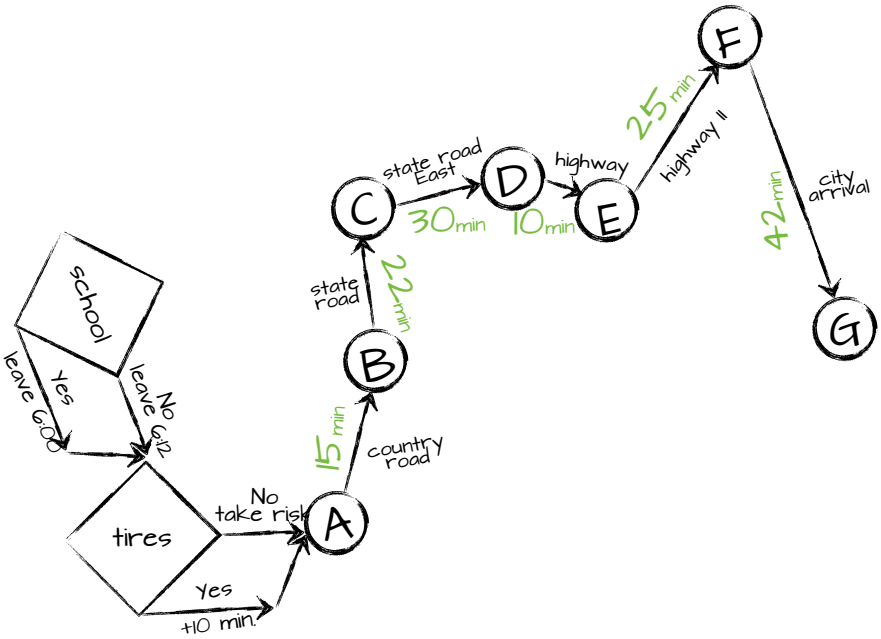
Fig. 3: Events.



Source: Author's design.

After consideration of all possible events, he adds travel time to each stage of the route. These estimations are based on memory or other sources like GPS. Final route sketching is on Fig. 4. Until now, this is only very simple logistic problem. But real life is, in most situations, not based on our simple planning and requires some amount of probability. Even with some time reserves, one or two unexpected events like traffic jam or accident might slow down Dave and cause late arrival. We will use Monte Carlo method to add some level of randomness to this model and run calculations many times to get probability of our successful route.

Fig. 4: Full sketch of route with estimated times.



Source: Author's design.

5. MONTE CARLO SIMULATION IMPLEMENTATION

Solution to the Morning meeting problem is quite simple with Monte Carlo Simulation. We just need to add some random probability to each stage of route. The estimated time en route is **144** minutes, without events. Our ETA (estimated time of arrival) might change because of the unpredictable events like fog, accidents, traffic etc. We set probability of up to 20% of being late. So each stage of route may last up to **20%** longer. On the other side, good weather, low traffic and other elements may impact our ETA positively, so we make get to the meeting sooner. For this situation, we choose up to probability **10%** of positive situation, so each stage of route may take as low as **90%** of planned time. It is up to decision maker to determine the optimal dispersion.

To make situation more real, we added events into the story. These are decision-making points, whether to take more time, to start earlier, to take or not to take risk etc. In the described situation, we have two events. Taking the child to school and checking tires. We can estimate the time required for each event. As with stages of route, time required for events can also extend or reduce. We set 96% - 106% variability of event time.

We cannot skip part of the route but we can skip event. Because of that, algorithm has 3 settings for events. First one means that we skip all events. The absolute opposite is third option, where we take all events. The second option is based of generated probability. There is **50%** chance of event happening.

Algorithm will take all these settings and will, with configured probability, calculate time required for trip. Because Monte Carlo simulation relies on the convergence of the mean result of a sequence of identical experiment, we run these calculations a lot of times with different random generated values and program returns us "confidence level" which indicates what is probability of getting to other city on time.

We run **5830** iterations of computation on **4** processors. **4652** successful on time arrives equals **0.7979** confidence level or **79%** probability or arriving on time.

5.1 Bigger simulation

To make things more interesting and parallel solution more visibly effective, we multiply number of route segments and events. Algorithm will generate all required data. For testing purposes, we use these settings:

- Number of Route segments: **80 000**
- Time reserve: **14%**
- Number of Events: **200**
- Number of iterations: **5 830 000**
- Length of one segment: **1 - 40**
- Length of one event: **1 - 10**
- Random variability of length of route segment: **90% - 125%**
- Probability of event: **50%**
- Random variability of length of event: **96% - 106%**

In this simulation, total number of calculations is **467 566 000 000**. Although each calculation is only simple multiplying and random number generation, the number of them is pushing us to use parallel computation. Parallelization was realized by simply splitting iterations between multiple processors. To preserve randomness, each processor has its own unique seed. After processors finish the calculations, first processor will gather results using **Reduce** method. The time required for calculations on different processor configurations is in Tab. 1.

Tab. 1: Speed-up of parallel algorithm.

Number of procesors	1	20
Time of calculation	6 hours 27 minutes	42,2 minutes
Speed-up	--	9,21

Source: Author's research.

6. CONCLUSION

Although all mentioned examples of Monte Carlo simulation are very simple, they clearly show that Monte Carlo Simulation is a very potent tool. One of the biggest advantages is simple parallelization and great speed-up. Some Monte Carlo Simulations are so complex that they require few days to calculate enough iterations to get required accuracy of confidence.
Monte Carlo simulation is used in different fields. For example in Physical sciences, Engineering (in wind energy yield analysis), computational biology, computer graphics, applied statistics (to compare competing statistics for small samples under realistic data conditions), artificial intelligence for games, finance and business etc.

References

1. BUFFON, G. Essai d'arithmétique morale. In *Histoire naturelle, générale er particuli`ere. Supplément 4.* 1777, pp. 46-123.
2. BUFFON, G. Solution des problµemes qui regardoient le jeu de franc carreau. In *Histoire de l'Académie royale des sciences.* 1733, pp. 43-45. ISSN 19674783.
3. JERONYMO, A. Running Monte Carlo Simulations in PHP. In *Sitepoint PHP.* [online] 2013. Collingwood: SitePoint Pty Ltd. [cit. 22.10.2013] Available on the internet: http://www.sitepoint.com/running-monte-carlo-simulations-in-php/
4. KALOS, M. H. – WHITLOCK, P. A. *Monte Carlo Methods.* German: Wiley-VCH, 2008. ISBN 978-3-527-40760-6.
5. KOCHANSKI, G. Monte Carlo Simulation. [online] 2005. [cit. 22.10.2013] Available on the internet: http://kochanski.org/gpk/teaching/0401Oxford/MonteCarlo.pdf
6. MATSUMOTO, M. – NISHIMURA. T. Mersenne Twister: A 623-Dimensionally Equidistributed Uniform Pseudo-Random Number Generator. In *ACM Transactions on Modeling and Computer Simulation.* 1998, vol. 8, No. 1, pp. 3-30.
7. RIPLEY, B. D. *Stochastic Simulation.* New Jersey: John Wiley & Sons, Inc., 1987, 237 p. ISBN 9780471818847.
8. SAWILOWSKY, S. S. You think you've got trivials? In *Journal of Modern Applied Statistical Methods.* 2003, vol. 2, No. 1, pp. 218-225. ISSN 1538–9472.