Solving implementation issues of Permutation Monte Carlo in different network reliability contexts *

Héctor Cancela
Facultad de Ingeniería, Universidad de la República
Montevideo, Uruguay
cancela@fing.edu.uy

and

Leslie Murray
FCEIA, Universidad Nacional de Rosario
Rosario. Argentina
lmurray@fceia.unr.edu.ar

and

Gerardo Rubino
INRIA–Rennes, Bretagne Atlantique, Campus de Beaulieu
Rennes, France
Gerardo.Rubino@inria.fr

Abstract

Network reliability computation is an NP-hard problem which has attracted much attention in literature. This problem consists in, given a network where the links may fail or operate with known probabilities, to compute the probability that a given subset of nodes (known as terminals) are connected by the operational links. Given the difficulty to compute the exact value of the network reliability, an alternative which has been much explored in the literature is the use of Monte Carlo estimation methods. In this work we discuss Permutation Monte Carlo, a highly efficient network reliability estimation method. The method is prone to numerical limitations for networks with a high number of links. We discuss this situation and present a simple way to rewrite the algorithm's calculations to make it numerically more stable. We also present a variant of Permutation Monte Carlo implementation for the particular case where all the links in the network under study share the same failure probability distribution (homogeneous network). For this type of network the Permutation Monte Carlo implementation can be redesigned to be much more efficient. We present some experimental test results proving that both proposed implementation variants are extremely efficient.

Keywords: Monte Carlo simulation, network reliability, creation process, numerical methods.

*This research was partially funded by the PEDECIBA program (Programa de Desarrollo de las Ciencias Básicas - Uruguay), and by Math-AMSUD project 19-MATH-03 RareDep.

1 Introduction

In recent years, network models have been an important support for the study of a wide variety of problems. The dynamic or static nature, the size, topology and type of connectivity, as well as the different definitions of reliability, make each network model appropriate for a different context.

The present work is based on the so called Network Static Model, a model designed for the analysis of connectivity problems. In this model, the network must guarantee the existence of paths to allow the connection between a set of nodes called terminal nodes. The links of the network are subject to failure, while the nodes are supposed to be perfect (this model can also be extended to include nodes failures). The failure of a link consists of its total interruption and occurs with known probability. Then, there is a probability —called reliability— that the links not failed form paths that guarantee the connection between the terminal nodes.

The computation of the reliability thus defined belongs to the class of NP-hard problem (see [1] and [2]). As all known algorithms have exponential complexity, for medium to large sized networks it is impossible to compute this measure in reasonable times [3, 4].

One proposal to avoid this problem is to build an estimate, using Monte Carlo methods, instead of computing the exact value of network reliability. However, as the links' reliability grow up to very high values and, therefore, the whole network becomes more reliable, the most direct Monte Carlo simulation approach, which is Standard Monte Carlo (Standard MC), becomes less efficient (see for instance [5] for a discussion). This is because, in a high reliability scenario, almost no replication will find the network in a state of failure while, in almost all of them the network will be in a perfect working condition. This causes an increase in the variance of the estimator. We can also justify this problem more formally as follows.

A parameter that is commonly used to evaluate the efficiency of a Monte Carlo estimate, a sort of relative error, is the Coefficient of Variation (CV), defined as the standard deviation divided by the estimated value. Suppose that the unreliability of the network under study (an extremely low value for a highly reliable network) is Q, then: $CV = (Q(1-Q))^{1/2}/Q$. Note that the denominator goes to zero faster than the numerator, causing the quotient to grow boundless as Q decreases.

Over the years, a large amount of research has aimed to obtain reliability estimators with lower variance than that of the Standard MC estimator on the Static Network Model. There is a huge literature on this subject, ranging from methods exploiting topological or probabilistic properties such as [6], [7], [8], [9], [10], [11] and [12] up to methods which first transform the original problem into a dynamic model and then perform the computations in this modified setting, such as [13], [14], [15], [16], [17], [18]. Also many papers discuss other network variants, including aspects such as distance constraints [19] and dependencies between states [20]. Some state of the art papers can be found at [21], [22], and [23]. Nevertheless, none of these works have addressed the stability of the methods from a computational, numerical analysis viewpoint (as opposed as from a probabilistic/statistical viewpoint).

One of the methods with good statistical properties is Permutation Monte Carlo, proposed in [14]. Nevertheless, the direct implementation of this method runs into numerical problems for large networks, which has precluded its application in these contexts.

In this work, we discuss what is the source of this numerical instability, and we propose a simple way to solve it by rewriting a key algorithmic component of the method. At the end of the article we include some experimental tests that demonstrates the benefits of the proposed modifications.

We also analyze the homogeneous network model, a specific variant in which all the links fail (or do not fail) with the same probability. The fact that the links share the same failure probability results in a simplification that some methods exploit to become more efficient. Although most networks correspond to the most general case, that is, non–homogeneous models, in many cases they fit the homogeneous model. For this reason, and because there are interesting methods for this latter case, the homogeneous model is worth analyzing. In particular, an important modification can be made to Permutation Monte Carlo in order to make it much more efficient when applied over homogeneous models. We will introduce this variant in Section 7. Within this context we present F–Monte Carlo in Section 8. While most methods in literature are designed for non-homogeneous networks, F–Monte Carlo has been designed to operate exclusively on homogeneous models, which allows it to attain particularly high performance levels in that setting [24]. This high efficiency makes it particularly interesting as a benchmark method. In the last part of the article we present some experimental tests in which we compare our new implementation of Permutation Monte Carlo against F–Monte Carlo.

This article is an extension of a preliminary work presented at CLEI 2024 under the title "Robust Implementation of Permutation Monte Carlo For Network Reliability Estimation" [25], which discussed the numerical problems of Permutation Monte Carlo in the heterogeneous case. In the present work, we add the discussion of the homogeneous model and new experimental tests.

The work is organized as follows. In Section 2 we present the Network Static Model. Sections 3, 4 and 5

discuss the Permutation Monte Carlo method and the numerical instabilities in its usual implementations. In Section 6 we present our proposal for avoiding this stability by rearranging terms in a key computation of the method. Section 7 introduces the homogeneous network model and in Section 8 we present a brief review of F–Monte Carlo. Section 9 presents some numerical results, showing that the method is able to estimate reliability values for networks with hundreds of components. Finally, Section 10 discusses conclusions and future work.

2 Network Static Model

The Network Static Model is a graph $G = (V, E)$ in which a set of nodes, V, is connected by a set of links, E, with $|V| = n$ and $|E| = m$. Nodes are perfect, only links fail, and they do it independently from each other. Links are modeled by the random vector $\mathbf{X} = (X_1, \ldots, X_m) \in \Omega = \{0, 1\}^m$, where $X_i = 0$ means that link i is failed (we say *down*), and is equivalent to removed from the graph, whereas $X_i = 1$ means that link i is operational (we say *up*). We say that vector $\mathbf{X}$ is the state of the network and we call configuration, denoted $\mathbf{x} = (x_1, \ldots, x_m)$, to any possible value of $\mathbf{X}$.

The network state space Ω is the union of two disjoint subspaces, one of which associated to a network *up* condition, and the other one associated to a network *down* condition. Usually, network *up* means that a subset $K \subseteq V$ of nodes is connected by means of the *up* links, and network *down* means that, despite the *up* links, the subset $K \subseteq V$ is disconnected. These two conditions are reflected by the structure function $\phi(\mathbf{X}) \in \{0, 1\}$ as follows: $\phi(\mathbf{X}) = 0$ means that the network is *down*, whereas $\phi(\mathbf{X}) = 1$ means that the network is *up*.

In the most general case, $\mathbb{P}\{X_i = 1\} = p_i$ and $\mathbb{P}\{X_i = 0\} = q_i = 1 - p_i$, $i = 1, \ldots, m$, where the p_i value may be different for each different link i.

We thus define the network reliability, R, as follows:

$$R = \mathbb{P}\{\phi(\mathbf{X}) = 1\}, \tag{1}$$

and the unreliability, Q, as

$$Q = \mathbb{P}\{\phi(\mathbf{X}) = 0\} = 1 - R. \tag{2}$$

While there are different closed formulas to express R and Q, many of which seem simple to compute, all of them involve an exponential number of terms, so that their direct computation is out of reach, except for very small networks. As mentioned in Section 1, well known theoretical results classify exact reliability computation as an NP-hard problem (actually a #P-hard problem, see [1] and [2]), so, unless $P = NP$, there is no hope of finding a polynomial method for performing this computation.

3 Dynamic Models

The difficulty in the exact computation of the reliability leads to the search for estimators, the best being those that allow a precise result with low calculation effort. Hence, in this context, one of the most important requirements for an estimator —ultimately, a random variable— is its low variance.

One of the ways to address the problem of obtaining a low variance estimator is to transform the static model into a dynamic one, provided that both models have the same reliability, and making the estimation on the latter one. Such transformation is the basis of different methods, like Splitting and Permutation Monte Carlo, originally intended to operate over dynamic models. An extensive review of these ideas can be found in [14], [26], [27], [28], [29], [30] and [31].

Permutation Monte Carlo is based on considering time t, along which the links are *repaired* (they go from *down* to *up*), provided that they are *down* at time $t = 0$. This is the guiding idea of many important research lines, some of which have been also adapted to the case of multiple states per component, like the work in [31]. An original and efficient approach, one of whose applications can be found in [32], is based on a distribution called D–spectrum, proposed to make a particular description of the sequences of *repairs*, for the case of homogeneous systems. For a complete survey of these methods, readers are advised to look over [33] and [34].

Permutation Monte Carlo consists of assigning a probability distribution to the links' *repair* times, and analyzing the distribution of the instant at which the subset of nodes in K becomes connected.

4 Permutation Monte Carlo (PMC)

In PMC the Network Static Model is transformed into a dynamic model by the so called *Creation Process* (CP), proposed in [14]. At this transformation, the network state $\mathbf{X} = (X_1, \ldots, X_m)$, becomes the stochastic process $\mathbf{X}(t) = (X_1(t), \ldots, X_m(t)) \in \Omega = \{0, 1\}^m$, in which:

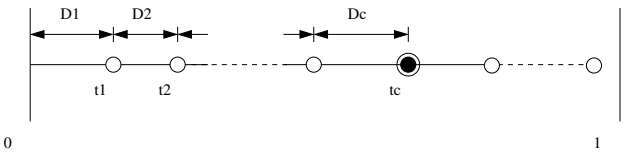

Figure 1: The Creation Process (CP)

$$X_i(t) = \begin{cases} 0 & \text{if } t < \tau_i \\ 1 & \text{otherwise,} \end{cases} \qquad i = 1, \ldots, m. \tag{3}$$

In words, at $t = 0$ all the links are *down*, or simply do not exist, while at times τ_i link i is repaired or created (it goes *up*). Every τ_i is exponentially distributed: $\tau_i \sim \text{Exp}(\lambda_i)$, with $\lambda_i = -\ln(q_i)$, then:

$$\mathbb{P}\{X_i(1) = 1\} = \mathbb{P}\{\tau_i \leq 1\} = 1 - e^{\ln(q_i)} = 1 - q_i = r_i \tag{4}$$

$$\mathbb{P}\{X_i(1) = 0\} = \mathbb{P}\{\tau_i > 1\} = e^{\ln(q_i)} = q_i \tag{5}$$

This means that, to observe CP at $t = 1$ is the same as observing the Network Static Model.

If at $t = 1$ all the links are *up* with probability equal to its own individual reliability, then the probability that the network is *up* at $t = 1$ is R. Similarly, the probability that the network is *down* at $t = 1$ is Q. Finally, $R = \mathbb{E}\{\phi(\mathbf{X}(1))\}$ and $Q = \mathbb{E}\{1 - \phi(\mathbf{X}(1))\}$.

There are $m!$ different ways of ordering the links' repairs, each of which is a *permutation* of the set $T = \{\tau_1, \tau_2, \ldots, \tau_m\}$, where τ_i is the repairing time of link i, $i = 1, \ldots, m$. Calling Π to the set of all the *permutations* in T, the general form of some $\pi \in \Pi$ is:

$$\pi = \{\tau_{(1)}, \tau_{(2)}, \ldots, \tau_{(m)}\}, \tag{6}$$

Figure 1 shows a possible sequence of repairs. It is important to distinguish the reference to the order in the sequence (the index in brackets) and the number of link repaired, because time $\tau_{(i)}$ is the repair time of, say, the j–th link, but not necessarily $i = j$. Actually, the terms $\tau_{(i)},\ i = 1, \ldots, m$, are the order statistics of T. The most important value in this set is $\tau_{(c)}$, the time at which $\phi(\mathbf{X}(t))$ goes to 1, that is:

$$\phi(\mathbf{X}(t)) = \begin{cases} 0 & \text{if } t < \tau_{(c)} \\ 1 & \text{otherwise.} \end{cases} \tag{7}$$

Thus, the event $\phi(\mathbf{X}) = 0$ in the Network Static Model is the same as event $\tau_{(c)} \geq 1$ in CP which, in the end, means that $Q = \mathbb{P}\{\tau_{(c)} \geq 1\}$. Then, given the following indicator variable:

$$I_P = \begin{cases} 1 & \text{if } \tau_{(c)} \geq 1 \\ 0 & \text{otherwise,} \end{cases} \tag{8}$$

we can build a crude or standard unreliability estimation, $\widehat{Q}_S$, based on the CP, as follows:

$$\widehat{Q}_S = \frac{1}{N} \sum_{i=1}^{N} I_P^{(i)}, \tag{9}$$

where $I_P^{(i)},\ i = 1, \ldots, N$, is a set of N independent copies of variable I_P.

In order to improve the estimation in (9), it is important to find a mathematical description of the time $\tau_{(c)}$. Considering the times between repairs, as shown in Figure 1, time $\tau_{(c)}$ is composed as follows:

$$\tau_{(c)} = \sum_{i=1}^{c} \Delta_i \tag{10}$$

where the times Δ_i can be computed, due to the properties of the exponential distribution.

Time Δ_1 is the smallest (earliest) out of a set of m exponentially distributed times, each one of them with rate $\lambda_i,\ i = 1, \ldots, m$. So, the rate of Δ_1 is $\Lambda_1 = \sum_{i=1}^{m} \lambda_i$, whereas the link repaired at time $\tau_{(1)}$

can be sampled from a discrete distribution in which the probability of each link is $\lambda_i/\Lambda_1,\ i = 1, \ldots, m$. Assuming that link z is sampled at time $\tau_{(1)}$, Δ_2 is exponentially distributed with rate $\Lambda_2 = \Lambda_1 - \lambda_z$, and the link repaired at time $\tau_{(2)}$ follows a discrete distribution in which the probability of each link is $\lambda_i/\Lambda_2\ \ i = 1, \ldots, m,\ i \neq z$.

Calling $\lambda_{(i)}$ to the rate of the exponential corresponding to the link repaired at time $\tau_{(i)}$, we can present the definition of the variables Δ_i as follows:

$$\Delta_1 \sim \text{Exp}(\lambda_{(1)} + \lambda_{(2)} + \ldots + \lambda_{(c)} + \ldots + \lambda_{(m)}) \tag{11}$$
$$\Delta_2 \sim \text{Exp}(\qquad\quad \lambda_{(2)} + \ldots + \lambda_{(c)} + \ldots + \lambda_{(m)}) \tag{12}$$
$$\vdots \tag{13}$$
$$\Delta_c \sim \text{Exp}(\qquad\qquad\qquad\quad \lambda_{(c)} + \ldots + \lambda_{(m)}) \tag{14}$$

The random variables $\Delta_1, \ldots, \Delta_c$, and, therefore the variable $\tau_{(c)} = \sum_{i=1}^{c} \Delta_i$, are determined by the permutation $\pi \in \Pi$. Thus, according to the Total Probability Theorem, the network unreliability can be expressed as:

$$Q = \mathbb{P}\{\tau_{(c)} \geq 1\} = \sum_{\pi} \underbrace{\mathbb{P}\{\tau_{(c)} \geq 1 \mid \Pi = \pi\}}_{\gamma(\pi)} \mathbb{P}\{\Pi = \pi\} \tag{15}$$

This expression is the basis of PMC which, in the end, is an application of a more general method known as Conditional Monte Carlo. The term $\gamma(\pi)$ is the probability that the network is up at $t = 1$ when the links are repaired according to the permutation π. Thus, if for any sampled permutation, the probability $\gamma(\pi)$ can be computed, the following Monte Carlo estimation is possible:

$$\widehat{Q}_P = \frac{1}{N} \sum_{i=1}^{N} \gamma(\pi^{(i)}), \tag{16}$$

where $\pi^{(i)},\ i = 1, \ldots, N$, is a set of N independent replications of the permutation π. It is important to notice that, in order to compute $\gamma(\pi^{(i)})$, it is not necessary to sample the times $\Delta_1, \ldots, \Delta_c$, for every replication, but only the order of repairs (permutation).

Next we show an analysis of the random variables underlying each of the two recently introduced estimators. In the case of PMC we have:

$$Q_P = \sum_{\pi} \gamma(\pi)\, \mathbb{P}\{\Pi = \pi\}, \tag{17}$$

from which it is simple to infer that:

$$\mathbb{V}\{Q_P\} = \sum_{\pi} \gamma(\pi)^2\, \mathbb{P}\{\Pi = \pi\} - \left(\sum_{\pi} \gamma(\pi)\, \mathbb{P}\{\Pi = \pi\} \right)^2. \tag{18}$$

Now we address the variance of Q_S, the random variable underlying the standard estimator:

$$\mathbb{V}\{Q_S\} = Q\,(1 - Q) \tag{19}$$
$$= \sum_{\pi} \gamma(\pi)\, \mathbb{P}\{\Pi = \pi\} \left(1 - \sum_{\pi} \gamma(\pi)\, \mathbb{P}\{\Pi = \pi\} \right) \tag{20}$$
$$= \sum_{\pi} \gamma(\pi)\, \mathbb{P}\{\Pi = \pi\} - \left(\sum_{\pi} \gamma(\pi)\, \mathbb{P}\{\Pi = \pi\} \right)^2 \pm \sum_{\pi} \gamma(\pi)^2\, \mathbb{P}\{\Pi = \pi\} \tag{21}$$
$$= \sum_{\pi} \gamma(\pi)^2\, \mathbb{P}\{\Pi = \pi\} - \left(\sum_{\pi} \gamma(\pi)\, \mathbb{P}\{\Pi = \pi\} \right)^2 + \sum_{\pi} \gamma(\pi)\, \mathbb{P}\{\Pi = \pi\} - \sum_{\pi} \gamma(\pi)^2\, \mathbb{P}\{\Pi = \pi\} \tag{22}$$
$$= \mathbb{V}\{Q_P\} + \underbrace{E(\gamma(\pi)) - E(\gamma(\pi)^2)}_{\geq 0} \tag{23}$$

what proves that $\mathbb{V}\{Q_P\} \leq \mathbb{V}\{Q_S\}$ and, ultimately, shows that the PMC estimation is more efficient than standard estimation.

5 PMC Implementation Issues

Once a permutation, π, is sampled, $\gamma(\pi)$ can be computed according to the distribution determined in [35], that in this context reads:

$$\mathbb{P}\{\Delta_1 + \ldots + \Delta_c > t \mid \pi\} = \tag{24}$$

$$\left(\prod_{i=1}^{c} \Lambda_i\right) \sum_{j=1}^{c} \frac{e^{-\Lambda_j\, t}}{\Lambda_j \prod\limits_{\substack{k=1 \\ k\neq j}}^{c} (\Lambda_k - \Lambda_j)} \tag{25}$$

For $t = 1$ this expression becomes:

$$\mathbb{P}\{\Delta_1 + \ldots + \Delta_c > 1 \mid \pi\} = \tag{26}$$

$$\left(\prod_{i=1}^{c} \Lambda_i\right) \sum_{j=1}^{c} \frac{e^{-\Lambda_j}}{\Lambda_j \prod\limits_{\substack{k=1 \\ k\neq j}}^{c} (\Lambda_k - \Lambda_j)} \tag{27}$$

The use of this formula is affected by both, the number of links and their reliability. As an numerical example, if $q_i = 1 \times 10^{-6}$, a typical value in high–reliability benchmark models, we have $\lambda_i = 13.8$, whereas for $q_i = 1 \times 10^{-8}$, $\lambda_i = 18.4$. See also that every Λ_i is the sum of many —eventually very many— rates λ_i. Thus, the product of the terms Λ_i may become large —eventually extremely large—.

Besides, the product of the terms $(\Lambda_k - \Lambda_j)$ have alternating signs, + and −. Globally, we add the results of these products, many of which are extremely large, with alternating sign, to produce a final result that is positive and large.

On the other hand, we have the term $e^{-\Lambda_j}$. Think, for example, of some Λ_j composed of ten identical values like $\lambda_i = 13.8$. This makes $e^{-138} = 1.17 \times 10^{-60}$. If, instead, we consider twenty values of λ_i equal to 18.4, we have $e^{-368} = 1.51 \times 10^{-160}$.

In summary, this formula deals with numbers, some that are extremely large and some others that are extremely small. When problems like this occur, the precision of the computer programs may not always be enough to avoid instability and numerical errors produced by overflows/underflows (either on too large or too small numbers). This is one reason why PMC, whose mechanism strongly rely on this formula, loses efficiency in the case of large network models.

A detailed analysis of this formula is addressed in [20] where the authors consider alternatives, such as representing the distribution as a matrix exponential. This solves the instability of the calculations at the cost of a significant increase in the complexity of the algorithm, making it clear that the drawbacks of the formula are difficult to avoid. As part of previous work, the authors of that paper developed a recursive algorithm that computes the distribution as a convolution, but it is extremely slow and consumes high amounts of memory.

Another possibility for solving the computation of $\gamma(\pi)$ is to avoid its exact calculation and settle for an estimation. The work in [36] is a reference to this approach.

Let us now introduce a more detailed and illustrative example. Think of a 20 links network with $p_i = 0.9998,\ i = 1, \ldots, 20$, and see what happens on a replication sample for which $c = 15$:

$$\mathbb{P}\{\Delta_1 + \ldots + \Delta_c > 1 \mid \pi\} = \sum_{j=1}^{c} \underbrace{\frac{\left(\prod\limits_{i=1}^{c} \Lambda_i\right) e^{-\Lambda_j}}{\Lambda_j \prod\limits_{\substack{k=1 \\ k\neq j}}^{c} (\Lambda_k - \Lambda_j)}}_{S_j} \tag{28}$$

$$
\begin{array}{lcl}
\Lambda_1 = \texttt{170.34} & & S_1 = \texttt{ 1.22E-70} \\
\Lambda_2 = \texttt{161.83} & & S_2 = \texttt{-8.98E-66} \\
\Lambda_3 = \texttt{153.31} & & S_3 = \texttt{ 3.08E-61} \\
\Lambda_4 = \texttt{144.79} & & S_4 = \texttt{-6.53E-57} \\
\Lambda_5 = \texttt{136.28} & & S_5 = \texttt{ 9.54E-53} \\
\Lambda_6 = \texttt{127.76} & & S_6 = \texttt{-1.02E-48} \\
\Lambda_7 = \texttt{119.24} & \rightarrow & S_7 = \texttt{ 8.17E-45} \\
\Lambda_8 = \texttt{110.72} & & S_8 = \texttt{-5.03E-41} \\
\Lambda_9 = \texttt{102.21} & & S_9 = \texttt{ 2.38E-37} \\
\Lambda_{10} = \texttt{93.69} & & S_{10} = \texttt{-8.67E-34} \\
\Lambda_{11} = \texttt{85.17} & & S_{11} = \texttt{ 2.38E-30} \\
\Lambda_{12} = \texttt{76.65} & & S_{12} = \texttt{-4.82E-27} \\
\Lambda_{13} = \texttt{68.14} & & S_{13} = \texttt{ 6.77E-24} \\
\Lambda_{14} = \texttt{59.62} & & S_{14} = \texttt{-5.95E-21} \\
\Lambda_{15} = \texttt{51.10} & & S_{15} = \texttt{ 2.48E-18}
\end{array}
$$

$$
\mathbb{P}\{\Delta_1 + \ldots + \Delta_c > 1 \mid \pi\} = \texttt{1.26414E-16} \tag{29}
$$

Even when the network under analysis is rather small (for larger networks, the problem is much worse), some of the terms are extremely small and their representation, and also the operations in which they are involved —by means of a programming language— can result in precision losses and numerical problems. As a reference, the single-precision floating point representation usually employed in programming languages can only represent with full precision values greater or equal to 1.18×10^{-38} (and even allowing for precision losses, so-called sub-normal representation, only allows for numbers greater or equal to 1.4×10^{-45}). We can see that in this example there are many terms that are smaller than these values, and this then affects the Monte Carlo estimation.

An alternative would be to employ double precision floating numbers and arithmetic, which is available in most modern programming languages. Double precision representation can only represent with full precision values greater or equal to 2.23×10^{-308}. While this seems a very large range (and would indeed cover all the numbers discussed in the previous example), if we consider a similar example in which the network is composed of 90 links with $p_i = 0.9998,\ i = 1, \ldots, 90$, and we still consider the replication for which $c = 15$ we find, for instance, that $S_1 = 9.42 \times 10^{-318}$.

Think now of a network model composed of 130 links with $p_i = 0.9998,\ i = 1, \ldots, 90$. Here, for $c = 125$, we have $\prod_{i=1}^{c} \Lambda_i = 8.34 \times 10^{333}$ (in this case, exceeding the largest number that can be represented in double precision floating numbers, 1.80×10^{308}).

This shows that for medium-sized networks, with more than 50 links, even using double-precision arithmetic is not sufficient to correctly implement the PMC method.

6 Alternative Implementation for PMC

As we mentioned in the previous section, the main numerical instabilities for PMC appear when trying to compute $\gamma(\pi) = \mathbb{P}\{\Delta_1 + \ldots + \Delta_c > 1 \mid \pi\}$. We observe that it is possible to re-arrange the formula, to ensure

that neither very small nor very large terms appear:

$$\mathbb{P}\{\Delta_1 + \ldots + \Delta_c > 1 \mid \pi\} = \sum_{j=1}^{c} \frac{\left(\prod_{i=1}^{c} \Lambda_i\right) e^{-\Lambda_j}}{\Lambda_j \prod_{\substack{k=1 \\ k\neq j}}^{c} (\Lambda_k - \Lambda_j)} \tag{30}$$

$$= \sum_{j=1}^{c} \frac{\Lambda_j \prod_{\substack{i=1 \\ i\neq j}}^{c} \Lambda_i}{\Lambda_j \prod_{\substack{k=1 \\ k\neq j}}^{c} (\Lambda_k - \Lambda_j)} e^{-\Lambda_j} \tag{31}$$

$$= \sum_{j=1}^{c} \underbrace{\left(\prod_{\substack{i=1 \\ i\neq j}}^{c} \frac{\Lambda_i}{\Lambda_i - \Lambda_j}\right)}_{\alpha_j} e^{-\Lambda_j} \tag{32}$$

Even when the size of the terms Λ_i and $\Lambda_i - \Lambda_j$ may grow excessively, it is quite easy to accept that the size of the quotients α_j won't grow disproportionately and they will be perfectly manageable for a programming language:

$$\begin{aligned}
\alpha_1 &= \texttt{ 11628} \\
\alpha_2 &= \texttt{-171360} \\
\alpha_3 &= \texttt{ 1175720} \\
\alpha_4 &= \texttt{-4979520} \\
\alpha_5 &= \texttt{ 14549535} \\
\alpha_6 &= \texttt{-31039008} \\
\alpha_7 &= \texttt{ 49884120} \\
\alpha_8 &= \texttt{-61395840} \\
\alpha_9 &= \texttt{ 58198140} \\
\alpha_{10} &= \texttt{-42325920} \\
\alpha_{11} &= \texttt{ 23279256} \\
\alpha_{12} &= \texttt{-9405760} \\
\alpha_{13} &= \texttt{ 2645370} \\
\alpha_{14} &= \texttt{-465120} \\
\alpha_{15} &= \texttt{ 38760}
\end{aligned}$$

Let us now transform the terms $e^{-\Lambda_j}$ to allow proper operations with the terms α_j:

$$e^{-\Lambda_j} = e^{-\left(\lambda_{(j)} + \ldots + \lambda_{(c)} + \ldots + \lambda_{(m)}\right)} \tag{33}$$

$$= e^{-\lambda_{(j)}} \ \ldots \ e^{-\lambda_{(c)}} \ \ldots \ e^{-\lambda_{(m)}} \tag{34}$$

$$= e^{-(-\ln(q_{(j)}))} \ \ldots \ e^{-(-\ln(q_{(c)}))} \ \ldots \ e^{-(-\ln(q_{(m)}))} \tag{35}$$

$$= e^{\ln(q_{(j)})} \ \ldots \ e^{\ln(q_{(c)})} \ \ldots \ e^{\ln(q_{(m)})} \tag{36}$$

$$= q_{(j)} \ \ldots \ q_{(c)} \ \ldots \ q_{(m)} \tag{37}$$

Finally, the computation of $\mathbb{P}\{\Delta_1 + \ldots + \Delta_c > 1 \mid \pi\}$ can be solved by the following recurrence, up to $i = c - 1$:

$$\chi_{i+1} = q_{(i+1)}(\chi_i + \alpha_{i+1}) \tag{38}$$

starting by $\chi_1 = \alpha_1 q_{(1)}$, and setting to zero $\alpha_j,\ j = c+1, \ldots, m$. See this in a small example in which $c = 3$

for a network composed of 7 links:

$$\sum_{j=1}^{3} \alpha_j \, e^{-\Lambda_j} = \alpha_1 \, e^{-\Lambda_1} + \alpha_2 \, e^{-\Lambda_2} + \alpha_3 \, e^{-\Lambda_3} \tag{39}$$

$$= \alpha_1 \, q_{(1)} q_{(2)} q_{(3)} q_{(4)} q_{(5)} q_{(6)} q_{(7)} \tag{40}$$

$$+ \alpha_2 \, q_{(2)} q_{(3)} q_{(4)} q_{(5)} q_{(6)} q_{(7)} \tag{41}$$

$$+ \alpha_3 \, q_{(3)} q_{(4)} q_{(5)} q_{(6)} q_{(7)} \tag{42}$$

$$= q_{(3)} q_{(4)} q_{(5)} q_{(6)} q_{(7)} (\alpha_1 \, q_{(1)} q_{(2)} + \alpha_2 \, q_{(2)} + \alpha_3) \tag{43}$$

$$= q_{(3)} q_{(4)} q_{(5)} q_{(6)} q_{(7)} (q_{(2)} (\alpha_1 \, q_{(1)} + \alpha_2) + \alpha_3) \tag{44}$$

$$\tag{45}$$

Now, the values involved in the operations —that should be performed in the order determined by the recurrence— are the α_i, which are not too large, and the q_i which are small, but not extremely small. Thus, we can approach the target value through operations none of which involve numbers in a range that may cause a numerical problem.

7 PMC on Network Homogeneous Models

A particular case of the Network Static Model is that in which $\mathbb{P}\{X_i = 1\} = p$ and $\mathbb{P}\{X_i = 0\} = q = 1-p$, $i = 1, \ldots, m$. This type of model, where all the links share the same distribution, is called homogeneous model. There are algorithms designed to operate exclusively on homogeneous models, for example F–Monte Carlo [24] and D–Spectrum [31, 33]. Permutation Monte Carlo allows to significantly simplify its implementation when applied over homogeneous network models. Let us briefly recall the basics of Permutation Monte Carlo before we introduce this simplification.

In Figure 1 we see a possible sequence of repairs. In order to compute $\gamma(\pi) = \mathbb{P}\{\tau_{(c)} \geq 1 \mid \Pi = \pi\}$ by the formula (27), whether we use the calculation variants proposed in Section 5 or not, we need to identify the links repaired at every $\tau_{(i)}$, that is, the permutation π.

As explained in Section 4, π can be sampled as follows: the link repaired at time $\tau_{(1)}$ comes from a discrete distribution in which the probability of each link is λ_i/Λ_1, $i = 1, \ldots, m$, where $\Lambda_1 = \sum_{i=1}^{m} \lambda_i$. If the link repaired at time $\tau_{(1)}$ is z, the link repaired at time $\tau_{(2)}$ follows a discrete distribution in which the probability of each link is λ_i/Λ_2 $i = 1, \ldots, m$, $i \neq z$, where $\Lambda_2 = \Lambda_1 - \lambda_z$. And this continues until some link is repaired at time $\tau_{(c)}$.

Let us now apply these ideas to the an homogeneous model in which $\lambda_i = \lambda$, $i = 1, \ldots, m$. Now, $\Lambda_1 = \sum_{i=1}^{m} \lambda_i = m\,\lambda$, therefore $\tau_{(1)}$ is sampled from a discrete distribution in which the probability of each link is $\lambda/\Lambda_1 = 1/m$. Then, no matter which link would have been sampled first, the next link to be repaired comes from a discrete distribution in which the probability of each of the $m-1$ remaining links has probability $1/(m-1)$. This means that all the links can be successively sampled from uniform distributions, out of the set of links that have not yet been sampled.

See also that $\Lambda_i = (m - i - 1)\,\lambda$, $i = 1, \ldots, c$, what shows that Λ_i only depends on i, but not on which is the i–th repaired link. Many permutations share the same c, which is now the only necessary value to compute the probability $\mathbb{P}\{\tau_{(c)} \geq 1\}$. This significant simplification let us transform $\gamma(\pi)$ into $\gamma(c)$ as follows:

$$\gamma(c) = \mathbb{P}\{\Delta_1 + \ldots + \Delta_c > t \mid c\} \tag{46}$$

Thus, once a replication is sampled, the outcome of the formula (25) only depends on the number of links repaired, but not on which those links are. There are, therefore, m possible values for expression (46) —a many as possible values of c— which can be computed only once, at the beginning of the simulation. If we save these values into an array $P[c]$, $c = 1, \ldots, m$, the mechanism of PMC can be summarized as follows: create two sets, L_0 containing all the links with their variables in 0 and L_1 empty, then (i) sample uniformly one link from L_0, set its variable to 1 and move it into L_1, (ii) compute $\phi(\mathbf{X}_1)$, where $\mathbf{X}_1$ is the set of links in L_1 and if $\phi(\mathbf{X}_1) = 0$ go back to (i); otherwise make $c = |L_1|$ and return $P[c]$ as the outcome of the replication.

After performing N independent replications, we can make the unreliability estimation indicated in (16), replacing each $\gamma(\pi^{(i)})$ by the outcome of the algorithm introduced in the prior paragraph.

In short, when it operates over homogeneous models, the Permutation Monte Carlo algorithm can be simplified because of two reasons: firstly because in every new permutation each link is sampled from a uniform distribution, secondly —and this the main reason— because we can avoid the computation of Formula (27) in each replication.

8 F–Monte Carlo

The F–Monte Carlo method [24] is a simulation technique for estimating network unreliability that works exclusively over homogeneous models. It is based on an exact unreliability formula, in which some unknown, and difficult–to–calculate, coefficients are replaced by unbiased estimators.

In order to introduce the formula in which the method is based, it is necessary to partition the space $\Omega = \{0,1\}^m$ as follows:

$$\Omega_j = \left\{ \mathbf{x} \in \Omega \ : \ \sum_{i=1}^{m} x_i = m - j \right\} \quad j = 0, \ldots, m,$$

All the configurations in Ω_j are composed of j zeros and $m-j$ ones, therefore $\mathbb{P}\{\mathbf{X} = \mathbf{x} \mid \mathbf{x} \in \Omega_j\} = q^j\, p^{m-j}$. Since $\mid \Omega_j \mid = \binom{m}{j}$, we have:

$$\sum_{j=0}^{m} \binom{m}{j} q^j\, p^{m-j} = 1.$$

Let $\alpha_j \leq 1,\ j = 0, \ldots, m$, be the fraction of configurations $\mathbf{x} \in \Omega_j$ for which $\phi(\mathbf{x}) = 0$, then:

$$\sum_{j=0}^{m} \alpha_j \binom{m}{j} q^j\, p^{m-j} = Q. \tag{47}$$

There is only one configuration $\mathbf{x}$ in Ω_0, and for this configuration $\phi(\mathbf{x}) = 1$ (none of the links are *down*), therefore, $\alpha_0 = 0$. There is also only one configuration $\mathbf{x} \in \Omega_m$, and for this configuration $\phi(\mathbf{x}) = 0$ (all the links are *down*), therefore $\alpha_m = 1$. Based on this we can write:

$$Q = \sum_{j=1}^{m} \alpha_j \binom{m}{j} q^j\, p^{m-j} = q^m \ + \sum_{j=1}^{m-1} \alpha_j \binom{m}{j} q^j\, p^{m-j} \tag{48}$$

Note that only the coefficients $\alpha_j,\ j = 1, \ldots, m-1$, are unknown. F–Monte Carlo is a simulation method aimed at estimating the network unreliability Q, based on expression (48). It consists in replacing the coefficients α_j by the unbiased estimators $\widehat{\alpha}_j,\ j = 1, \ldots, m-1$. Replacements made, the network unreliability estimator reads:

$$\widehat{Q}_{\mathrm{F}} = \sum_{j=1}^{m} \widehat{\alpha}_j \binom{m}{j} q^j\, p^{m-j} = q^m \ + \sum_{j=1}^{m-1} \widehat{\alpha}_j \binom{m}{j} q^j\, p^{m-j}. \tag{49}$$

The idea for obtaining the estimators $\widehat{\alpha}_j$ is to make Crude Monte Carlo estimations in the subspaces $\Omega_j,\ j = 1, \ldots, m-1$.

Since the parameters α_j in (47) are just the fraction of configurations $\mathbf{x} \in \Omega_j$ for which $\phi(\mathbf{x}) = 0$, no matter the probability distribution of the links, we can make the estimations in every Ω_j using different probability values, other than p and q. In fact, sampling under p and q will produce a high number of configurations $\mathbf{x}$ for which $\phi(\mathbf{x}) = 1$, and a few or none for which $\phi(\mathbf{x}) = 0$, causing an increase in the variance of the estimators $\widehat{\alpha}_j$ and, therefore, in the variance of $\widehat{Q}_{\mathrm{F}}$.

To make efficient estimations, we may use the random variable $\mathbf{Y} = (Y_1, \ldots, Y_m) \in \Omega = \{0,1\}^m$, where $Y_i = 0$ means that link i is *down*, $Y_i = 1$ means that link i is *up* and $\mathbb{P}\{Y_i = 0\} = \mathbb{P}\{Y_i = 1\} = 0.5\ \forall\ i$. Based on $\mathbf{Y}$ we may partition the state space $\Omega = \{0,1\}^m$ as follows:

$$\Omega_j = \left\{ \mathbf{y} \in \Omega : \ \sum_{i=1}^{m} y_i = m - j \right\}, \quad j = 0, \ldots, m.$$

In every Ω_j we may define $\mathbf{Y}_j$, a random vector of size m, composed of j zeros and $m-j$ ones. Every $\mathbf{Y}_j$ is uniformly distributed with $\mathbb{P}\{\mathbf{Y}_j = \mathbf{y}_j \mid \mathbf{y}_j \in \Omega_j\} = 1/\binom{m}{j}$. As α_j is the fraction of configurations $\mathbf{x} \in \Omega_j$ for which $\phi(\mathbf{x}) = 0$, α_j is also the fraction of values of $\mathbf{y}_j$ for which $\phi(\mathbf{y}_j) = 0$.

To build the estimators $\widehat{\alpha}_j$ we may generate N_j samples of $\mathbf{Y}_j$. The process for obtaining these samples can be implemented as follows: (i) fill with 1s all the components of a vector $\mathbf{y}_j$, (ii) choose uniformly, and fill with 0s, j of the m components of $\mathbf{y}_j$. There is an algorithm called RPERM, proposed by Fishman in [37],

that implements this process very efficiently. Thus, if, for every j, we execute this algorithm N_j times, we can produce N_j samples, $\mathbf{Y}_j^{(i)},\ i = 1, \ldots, N_j$ and then build the estimator of α_j as follows:

$$\widehat{\alpha}_j = \frac{1}{N_j} \sum_{i=1}^{N_j} 1 - \phi\big(\mathbf{Y}_j^{(i)}\big), \quad j = 1, \ldots, m-1, \tag{50}$$

Once we have built these estimators, we have all we need to compute the network reliability estimator, $\widehat{Q}_{\mathrm{F}}$, introduced in (49).

9 Computational results

9.1 PMC Implementation Improvements

The purpose of the experiments presented in this section is twofold, first to quantify the performance improvement of PMC against the Standard MC method, and then to show the benefit of the methodology proposed in this article.

The Standard MC unreliability estimation, $\widehat{Q}_S$, is based on N independent copies of the variable $\mathbf{X}$, sampled from the original distribution: $\mathbb{P}\{X_i = 1\} = p$ and $\mathbb{P}\{X_i = 0\} = q = 1 - p$, $i = 1, \ldots, m$. According to this, the estimator is:

$$\widehat{Q}_S = 1 - \frac{1}{N} \sum_{i=1}^{N} \phi\left(\mathbf{X}^{(i)}\right) \tag{51}$$

The usual measure to compare a simulation method like PMC against the standard method is the *speedup* value. We call it here $\mathbb{W}_{S/P}$, and we compute it by the following quotient:

$$\mathbb{W}_{S/P} = \frac{t_S \times \mathbb{V}_S}{t_P \times \mathbb{V}_P} \tag{52}$$

where t_S and t_P are the simulation times of Standard MC and PMC, and $\mathbb{V}_S$ and $\mathbb{V}_P$ are the respective variances. This measure allows to take into consideration both, the improvement in precision obtained by the method under consideration (in this case PMC), and the eventual difference in computational effort between the methods.

First of all we present a case much used in literature, the Dodecahedron network (see [11], [18], [38], [39] and [40]), shown in Figure 2. With the same topology, we make five different experiments, corresponding to different individual link reliability values: $r_i = 0.9; 0.99; 0.999; 0.9999; 0.99999$; corresponding to stricter reliability values and more demanding and difficult numerical estimations. Table 1 rows correspond to the individual link reliabilities; the columns show the unreliability estimation computed by PMC, and the speedups over the Standard MC method. It is possible to observe that in the low-reliability case, Standard MC is a bit more efficient than PMC; but when reliability values grow, PMC has a striking advantage, achieving a speedup of many orders of magnitude. This is also shown in graphical format in Figure 3.

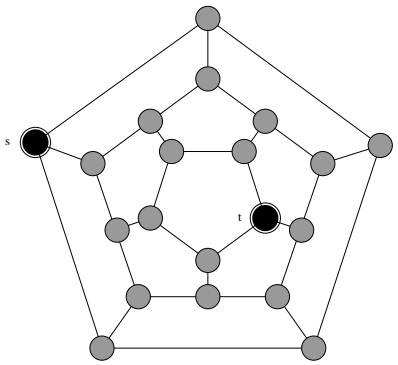

Figure 2: Dodecahedron Network

As a second test case we present the results obtained over the G–20 network. Figure 4 shows the general topology of G–n graphs. We have chosen the case where $n = 20$, corresponding to a network with 42 nodes and 82 links. Table 2 includes the following columns: individual link reliabilities, unreliability estimation

Table 1: PMC vs. Standard MC over Dodecahedron Network

r_i	$\widehat{Q}_P$	$\mathbb{W}_{S/P}$
0.9	2.87E−03	6.15E−01
0.99	2.03E−06	3.32E+01
0.999	1.97E−09	2.21E+04
0.9999	1.96E−12	2.14E+07
0.99999	1.96E−15	2.12E+10

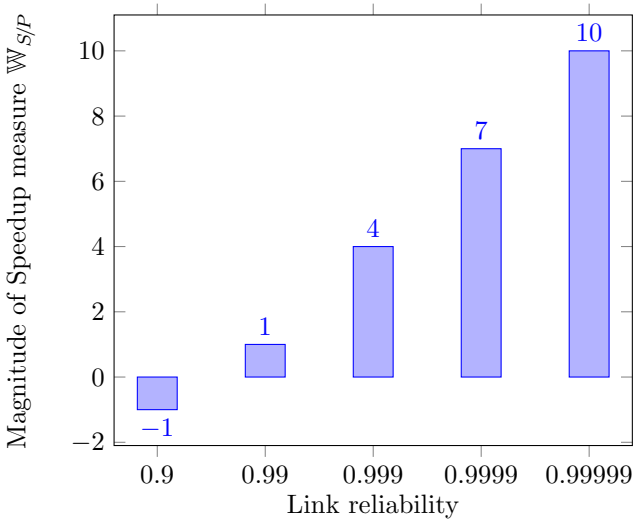

Figure 3: Bar chart of speedup order of magnitudes for the Dodecahedron network

Table 2: PMC vs. Standard MC over G–20 Network

r_i	$\widehat{Q}_P$	$\mathbb{W}_{S/P}$
0.9	5.75E−02	1.17E−01
0.99	2.43E−04	5.03E−01
0.999	2.07E−06	2.07E+01
0.9999	2.03E−08	1.85E+03
0.99999	2.03E−10	1.82E+05

computed by PMC, and speedup over the Standard MC method. Again we report five different experiments, corresponding to the link reliability values $r_i = 0.9; 0.99; 0.999; 0.9999; 0.99999$.

In third place we present results obtained over the Grid–$n \times n$ network (shown in Figure 5) for the case of $n = 5$. The Grid–5×5 network has 25 nodes and 40 links. The results are shown in Table 3, including again the same columns: individual link reliabilities, unreliability estimation computed by PMC, and speedup over the Standard MC method. Five different experiments are reported for each topology, corresponding to the link reliability values $r_i = 0.9; 0.99; 0.999; 0.9999; 0.99999$.

From the results reported in the three tables, we can see that our implementation of the PMC estimator is able to obtain good estimations for the cases of very reliable networks, which are a challenge for the Standard MC; without being hampered by the small numbers/potential underflow effect of the original PMC proposal. The results are consistently good for networks of different sizes.

Now we show, in Table 4, the last set of results from experiments designed to try out the main proposal in this article. On the left hand side of the table, shaded in gray, we have results that correspond to simulations carried out through the classic PMC implementation, that is, before the proposed improvements. On the right hand side of the table, we have simulations performed on the same network models, with a PMC implementation in which we include the modifications proposed to avoid numerical instabilities. Every experiment whose result is shown as ***** corresponds to a simulation affected by extremely large or extremely small numbers and, as a consequence, an error in the final unreliability estimation (the implementation stops with an exception error message, without giving a numerical estimate). The case of extremely large numbers comes up with the growth of the network in terms of the number of links, the extremely small numbers correspond to the increment of the individual link reliability. The Grid–$n \times n$ networks, for $n = 8$, $n = 9$

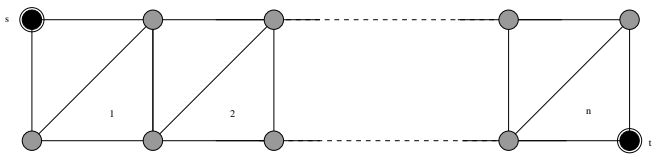

Figure 4: G–n Networks

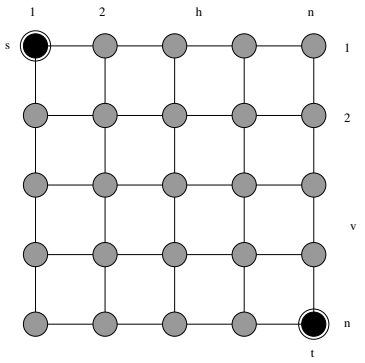

Figure 5: PMC vs. Standard MC over Grid–$n \times n$ Network

and $n = 10$ have, respectively, 112, 144 and 180 links.

The experiments were conducted on a DELL Inspiron 15 computer, processor Intel(R) Core(TM) i5-8250U, operating system Linux, kernel version 5.4.0-150. All the implementations were programmed in C language, compiler `gcc`, version 7.5.0.

The number of replications for the PMC executions in Tables 1, 2 and 3 were, respectively, 10×10^6, 1×10^6 and 4×10^6, whereas, the number of replications for the Standard MC simulations were 100×10^6, 20×10^6 and 50×10^6. These values were chosen so that all the simulations would run for approximately 100 seconds.

In the case of Table 4, in which the only purpose is to distinguish those implementations that work correctly from those that do not, a number of 10^5 replications per PMC run was enough.

From the results reported in the four tables, we can see that our implementation of the PMC estimator is able to obtain good estimations for the cases of very reliable networks, which are a challenge for the Standard MC; without being hampered by the small numbers/potential underflow effect of the original PMC proposal. The results are consistently good for networks of different sizes, up to 180 links, which is quite demanding computationally and numerically.

While the Standard MC has decreasing precision levels for more reliable graphs, we can see that PMC has a better performance, leading to increasing gains when compared against Standard MC. The algorithmic ideas behind PMC are stable even for very high link reliability values, and our implementation also removes numerical unstabilities, leading to these very good results.

9.2 PMC and F–Monte Carlo over Homogeneous Models

The experiments shown in this section focus on the PMC variant proposed for the homogeneous case called here H–PMC. Again we base the experimental comparison on the *speedup*, now defined as follows:

$$\mathbb{W}_{m/H} = \frac{t_m \times \mathbb{V}_m}{t_H \times \mathbb{V}_H} \tag{53}$$

where H refers to H–PMC and m to the method to which H–PMC is compared.

We run the simulations on the benchmark networks shown in Figures (6), (7), (8), and (9) which are composed as follows:

In Tables 5 and 6 we evaluate the efficiency increase of the proposed implementation for homogeneous systems. To do so, we estimate the unreliability twice, firstly using the standard PMC implementation with the improvements proposed in Section 6 ($\widehat{Q}_P$), secondly using the implementation proposed in Section 7 for homogeneous systems ($\widehat{Q}_H$). Since the proposed modification does not modify the accuracy, the improvement in *speedup* is only due to a reduction in the execution times. Specifically, the execution times of the improved

Table 3: PMC vs. Standard MC over Grid–5×5 Network

r_i	$\widehat{Q}_P$	$\mathbb{W}_{S/P}$
0.9	2.45E−02	2.01E−01
0.99	2.06E−04	1.55E+00
0.999	2.02E−06	9.97E+01
0.9999	2.01E−08	9.48E+03
0.99999	2.01E−10	6.95E+05

Table 4: $\widehat{Q}_P$ over Grid–$n \times n$ Networks

	Standard implementation			Proposed implementation		
r_i	$n = 8$	$n = 9$	$n = 10$	$n = 8$	$n = 9$	$n = 10$
0.9	2.45E−02	2.37E−02	*****	2.45E−02	2.37E−02	2.32E−02
0.99	2.34E−04	*****	*****	2.34E−04	1.80E−04	1.97E−04
0.999	2.52E−06	*****	*****	2.52E−06	1.57E−06	2.05E−06
0.9999	2.55E−08	*****	*****	2.55E−08	1.55E−08	2.09E−08
0.99999	2.55E−10	*****	*****	2.55E−10	1.54E−10	2.09E−10
0.999999	*****	*****	*****	2.55E−12	1.54E−12	2.09E−12

Network	Nodes	Links
Ring–1×16	16	16
Ring–2×16	32	48
Ring–3×16	48	80
Ring–4×16	64	112

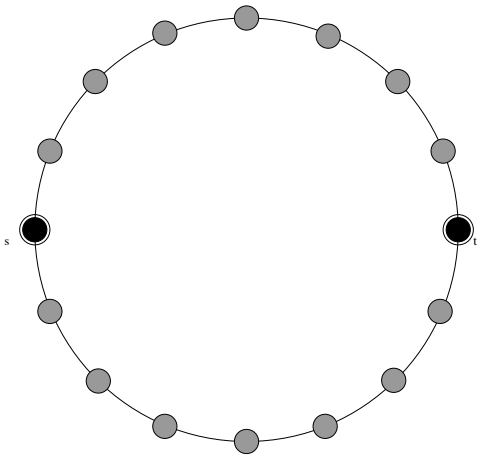

Figure 6: Ring–1×16 Network

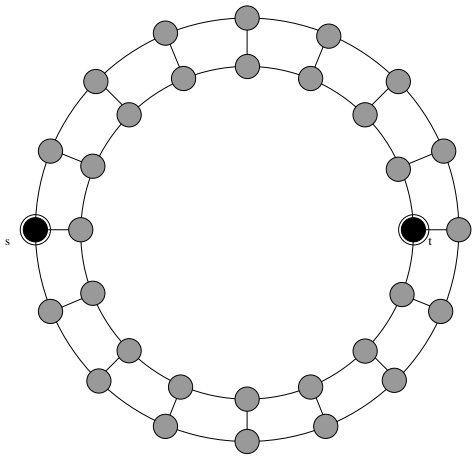

Figure 7: Ring–2×16 Network

variant are in the order of two thirds of the times of the standard variant. And this improvement grows, slightly, along with the growth of the reliability and the network size.

Table 5: PMC vs. H–PMC over Ring–2×16

r_i	$\widehat{Q}_P$	$\widehat{Q}_H$	$\mathbb{W}_{P/H}$
0.9	1.04E−02	1.04E−02	1.46
0.99	2.65E−06	2.63E−06	1.52
0.999	2.02E−09	1.98E−09	1.53
0.9999	1.95E−12	1.92E−12	1.53

In Tables 7 and 8 we show an experimental comparison between the variant of PMC suitable for homogeneous systems and F–Monte Carlo. The unreliability is estimated by both methods, $\widehat{Q}_H$ and $\widehat{Q}_F$, and the corresponding *speedup* $\mathbb{W}_{F/H}$ is evaluated. An improvement is observed which, as in the two previous

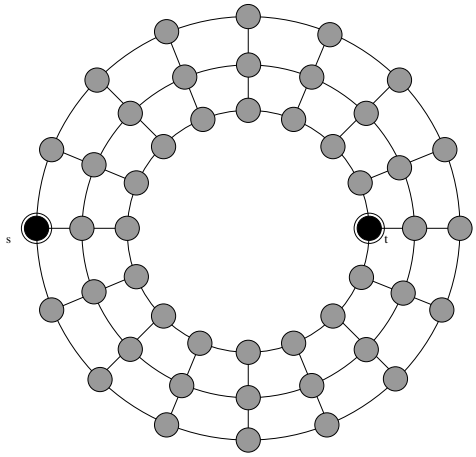

Figure 8: Ring–3×16 Network

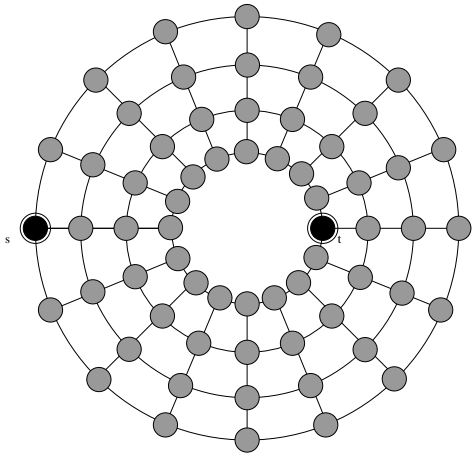

Figure 9: Ring–4×16 Network

Table 6: PMC vs. H–PMC over networks with $r_i = 0.999$

Network	$\widehat{Q}_P$	$\widehat{Q}_H$	$\mathbb{W}_{P/H}$
Ring–1×16	6.35E−05	6.36E−05	1.51
Ring–2×16	2.02E−09	1.98E−09	1.53
Ring–3×16	1.81E−09	1.69E−09	1.58
Ring–4×16	2.11E−09	1.81E−09	1.76

experiments, grows along with the growth of the network reliability and the network size.

Table 7: F–MC vs. H–PMC over Ring–3×16

r_i	$\widehat{Q}_F$	$\widehat{Q}_H$	$\mathbb{W}_{F/H}$
0.9	1.05E−02	1.04E−02	1.17
0.99	2.67E−06	2.63E−06	3.27
0.999	1.96E−09	1.98E−09	3.95
0.9999	1.88E−12	1.92E−12	4.01

Table 8: F–MC vs. H–PMC over networks with $r_i = 0.999$

Network	$\widehat{Q}_F$	$\widehat{Q}_H$	$\mathbb{W}_{F/H}$
Ring–1×16	6.35E−05	6.36E−05	2.01
Ring–1×16	1.96E−09	1.98E−09	4.04
Ring–1×16	1.91E−09	1.69E−09	6.22
Ring–1×16	2.00E−09	1.81E−09	7.46

10 Conclusions and future work

In this work, we discuss the PMC simulation method for estimating network reliability values, and we show that the original formulation involves the computation of both extremely small and extremely large values, which for large networks exceed usual floating point precision capacities. We present an alternative way to apply the method, by rewriting terms of a key formula, avoiding this issue and resulting in a more robust implementation. Computational experiments show that the improved PMC implementation proposal is able to estimate the reliability in medium and large sized networks where the standard implementation did not work, as numerical exceptions resulted in aborted executions. It is interesting to note that these problems arise even in cases where the link reliabilities and the overall network reliability are not extremely high, i.e, the problem is with the intermediate steps of the computations, and not with the expected final results. The numerical results, presented over a variety of network topologies of different sizes and structures, show the efficiency and robustness of the PMC method when implemented following our proposal.

We also show how to take advantage of some issues in the implementation of PMC that allow it to be adapted very efficiently to the homogeneous case, in which all links share the same probability distribution. We shown, experimentally, that the variant adjusted to this case is particularly efficient, outperforming, when tested on homogeneous networks, both the standard PMC and F–Monte Carlo, the latter being a method specifically designed for homogeneous networks.

While both implementations improve both the robustness and the efficiency of the PMC method, we do not have a theoretical result which allows to know in advance the boundaries in network size and link reliability values which guarantee robust computations. Future work can include studying both this issue, and also apply similar analysis to other simulation methods from literature . This aspect of the numerical implementation of the methods is not much studied in previous literature, and for practical applications it can be of relevance.

References

[1] A. Rosenthal, "Computing the reliability of complex networks," *SIAM Journal on Applied Mathematics*, vol. 32, no. 2, pp. 384–393, 1977.

[2] S. J. Provan and M. O. Ball, "The Complexity of Counting Cuts and of Computing the Probability that a Graph is Connected," *SIAM Journal on Computing*, vol. 12, no. 4, pp. 777–788, 1983.

[3] L. Schäfer, S. García, and V. Srithammavanh, "Simplification of inclusion-exclusion on intersections of unions with application to network systems reliability," *Reliability Engineering and System Safety*, vol. 173, pp. 23–33, 2018.

[4] K. Kobayashi, H. Morohosi, and T. Oyama, "Applying path-counting methods for measuring the robustness of the network-structured system," *International Transactions in Operational Research*, vol. 16, no. 3, pp. 371–389, 2009. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1475-3995.2008.00688.x

[5] G. Rubino, "Network reliability evaluation," in *State-of-the art in performance modeling and simulation*, K. Bagchi and J. Walrand, Eds. Gordon and Breach Books, 1998, ch. 11, pp. 275–302.

[6] H. Kumamoto, K. Tanaka, and K. Inoue, "Efficient evaluation of system reliability by Monte Carlo method," *IEEE Transactions on Reliability*, vol. R–26, no. 5, pp. 311–315, Dec. 1977.

[7] H. Kumamoto, K. Tanaka, K. Inoue, and E. Henley, "Dagger–sampling Monte Carlo for system unavailability evaluation," *IEEE Transactions on Reliability*, vol. R–29, no. 2, pp. 122–125, June 1980.

[8] G. Fishman, "A Monte Carlo sampling plan for estimating network reliability," *Operational Research*, vol. 34, 1986.

[9] S. M. Ross, "A new simulation estimator of system reliability," *Journal of Applied Mathematics and Stochastic Analysis*, vol. 7, no. 3, 1994.

[10] H. Cancela and M. El Khadiri, "A recursive variance–reduction algorithm for estimating communication–network reliability," *IEEE Transactions on Reliability*, vol. 44, no. 4, pp. 595–602, December 1995.

[11] ——, "The recursive variance–reduction simulation algorithm for network reliability evaluation," *IEEE Transactions on Reliability*, vol. 52, no. 2, pp. 207–212, June 2003.

[12] P. L'Ecuyer, G. Rubino, S. Saggadi, and B. Tuffin, "Approximate Zero-Variance Importance Sampling for Statistic Network Reliability Estimation," *IEEE Transactions on Reliability - TR*, vol. 60, no. 3, pp. 590–604, 2011.

[13] M. Easton and C. Wong, "Sequential destruction method for Monte Carlo evaluation of system reliability," *IEEE Transactions on Reliability*, vol. R–29, no. 1, April 1980.

[14] T. Elperin, I. B. Gertsbakh, and M. Lomonosov, "Estimation of network reliability using graph evolution models," *IEEE Transactions on Reliability*, vol. 40, no. 5, pp. 572–581, Dec 1991.

[15] M. Lomonosov, "On Monte Carlo estimates in network reliability," *Probability in the Engineering and Informational Sciences*, vol. 8, pp. 245–264, 1994.

[16] K.-P. Hui, N. Bean, M. Kraetzl, and D. Kroese, "The tree cut and merge algorithm for estimation of network reliability," *Probability in the Engineering and Informational Sciences*, vol. 17, no. 1, pp. 23–45, 2003.

[17] ——, "The cross–entropy method for network reliability estimation," *Annals of Operations Research*, vol. 134, pp. 101–118(18), February 2005.

[18] L. Murray, H. Cancela, and G. Rubino, "A splitting algorithm for network reliability," *IIE Transactions*, vol. 45, no. 2, pp. 177–189, 2013.

[19] E. Canale, F. Robledo, P. Romero, and P. Sartor, "Monte Carlo methods in diameter-constrained reliability," *Optical Switching and Networking*, vol. 14, Part 2, no. 0, pp. 134 – 148, 2014, special Issue on RNDM 2013.

[20] Z. I. Botev, P. L'Ecuyer, R. Simard, and B. Tuffin, "Static network reliability estimation under the Marshall-Olkin copula," *ACM Transactions on Modeling and Computer Simulation*, vol. 26, no. 2, p. Article 14, 2016.

[21] G. Fishman, "A comparison of four Monte–Carlo methods for estimating the probability of s–t connectedness," *IEEE Transactions on Reliability*, vol. R–35, no. 2, pp. 145–155, June 1986.

[22] H. Cancela, M. El Khadiri, and G. Rubino, "Rare events analysis by Monte Carlo techniques in static models," in *Rare Event Simulation using Monte Carlo Methods*, G. Rubino and B. Tuffin, Eds. John Wiley & Sons, 2009, ch. 7, pp. 145–170.

[23] H. Pérez-Rosés, "Sixty years of network reliability," *Mathematics in Computer Science*, vol. 12, no. 3, pp. 275–293, 2018.

[24] E. Canale, F. Robledo, P. Romero, and P. Sartor, "Monte Carlo methods in diameter-constrained reliability," *Optical Switching and Networking*, vol. 14, pp. 134–148, 2014, special Issue on RNDM 2013. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1573427714000691

[25] H. Cancela, L. Murray, and G. Rubino, "Robust implementation of Permutation Monte Carlo for network reliability estimation," in *Proceedings of the L Latin American Computer Conference (CLEI 2024), Bahia Blanca, Argentina*, 2024, pp. 1–8. [Online]. Available: https://ieeexplore.ieee.org/document/10700448

[26] P. Glasserman, P. Heidelberger, P. Shahabuddin, and T. Zajic, "Splitting for rare event simulation: Analysis of simple cases," in *Proceedings of the 28th conference on Winter simulation, WSC 1996, Coronado, CA, USA, December 8-11, 1996*, 1996, pp. 302–308. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/WSC.1996.873293

[27] M. J. J. Garvels, "The splitting method in rare event simulation," Ph.D. dissertation, Faculty of mathematical Science, University of Twente, The Netherlands, 2000.

[28] P. L'Ecuyer, V. Demers, and B. Tuffin, "Rare events, splitting, and Quasi-Monte Carlo," *ACM Trans. Model. Comput. Simul.*, vol. 17, no. 2, Apr. 2007. [Online]. Available: http://doi.acm.org/10.1145/1225275.1225280

[29] P. L'Ecuyer, F. Le Gland, P. Lezaud, and B. Tuffin, "Splitting techniques," in *Rare Event Simulation using Monte Carlo Methods*, G. Rubino and B. Tuffin, Eds. John Wiley & Sons, 2009, ch. 3, pp. 39–61.

[30] M. Amrein and H. R. Künsch, "A variant of importance splitting for rare event estimation: Fixed number of successes," *ACM Trans. Model. Comput. Simul.*, vol. 21, no. 2, pp. 13:1–13:20, Feb. 2011. [Online]. Available: http://doi.acm.org/10.1145/1899396.1899401

[31] I. Gertsbakh, R. Rubinstein, Y. Shpungin, and R. Vaisman, "Permutational methods for performance analysis of stochastic flow networks," *Probability in the Engineering and Informational Sciences*, vol. 28, no. 1, pp. 21–38, 2014.

[32] I. B. Gertsbakh, Y. Shpungin, and R. Vaisman, "D-spectrum and reliability of a binary system with ternary components," *Probability in the Engineering and Informational Sciences*, vol. 30, no. 1, p. 25–39, 2016.

[33] I. B. Gertsbakh and Y. Shpungin, *Models of Network Reliability: Analysis, Combinatorics, and Monte Carlo*, 1st ed. USA: CRC Press, Inc., 2009.

[34] I. B. Gertsbakh, Y. Shpungin, and R. Vaisman, *Ternary Networks - Reliability and Monte Carlo*, ser. Springer Briefs in Electrical and Computer Engineering. Springer, 2014. [Online]. Available: https://doi.org/10.1007/978-3-319-06440-6

[35] M. Balázs, "Sum of independent exponentials," Online notes - https://people.maths.bris.ac.uk/ mb13434/sumexp.pdf, 2005.

[36] I. Gertsbakh, E. Neuman, and R. Vaisman, "Monte Carlo for estimating exponential convolution," *Communications in Statistics - Simulation and Computation*, vol. 44, no. 10, pp. 2696–2704, 2015. [Online]. Available: https://doi.org/10.1080/03610918.2013.842591

[37] G. S. Fishman, *Monte Carlo : concepts, algorithms, and applications*, ser. Springer series in operations research. New York: Springer-Verlag, 1996.

[38] A. Moshnikov, "Evaluation of network reliability and element importance metrics," in *Majorov International Conference on Software Engineering and Computer Systems*, 2020. [Online]. Available: https://api.semanticscholar.org/CorpusID:235737006

[39] I. Gertsbakh, Y. Shpungin, and R. Vaisman, *Ternary Networks: Reliability and Monte Carlo*. Springer Publishing Company, Incorporated, 2014.

[40] S. Sebastio, K. S. Trivedi, D. Wang, and X. Yin, "Fast computation of bounds for two-terminal network reliability," *European Journal of Operational Research*, vol. 238, no. 3, pp. 810–823, 2014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0377221714003774