Robust implementation of Permutation Monte Carlo for network reliability estimation

Hector Cancela, Leslie Murray, Gerardo Rubino

Robust implementation of Permutation Monte Carlo for network reliability estimation

50th Conferencia Latinoamericana de Informática (CLEI'2024)

Best paper award

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

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

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. In this work, we discuss the Permutation Monte Carlo method, which is an estimation algorithm which has shown much promise but that is prone to numerical problems in the case of networks with a large number of links. We discuss this situation and we present a simple way to rewrite the algorithm's computations which is more numerically stable. We present some computational results showing that the method is more efficient that the standard Monte Carlo method for highly reliable networks, and that it can be applied to topologies with hundreds of links.

Index Terms

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

I. 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 links paths to allow the connection between a set of nodes called terminal nodes. Failures only affect links, not nodes. 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]) so, for medium to large sized networks, the exponential complexity of all known algorithms makes its computation impossible [3], [4]. One proposal to avoid this problem is, instead of trying to obtain an exact value, to compute an estimate of the reliability using Monte Carlo methods. However, when the network is highly reliable (i.e. when links' failure probabilities are extremely low), the standard Monte

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.

Carlo algorithm is not efficient, as obtaining a good estimate of the network failure probability becomes computationally expensive because its coefficient of variation grows boundlessly.

Over the years, a large amount of research has aimed to obtain reliability estimators with lower variance than that of the Standard Monte Carlo estimator on the Static Network Model. The following list includes some of the most relevant works on this subject: [5], [6], [7], [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21] and [22].

One of the methods with good statistical properties is Permutation Monte Carlo, proposed in [10]. 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.

The work is organized as follows. In Section II we present the Network Static Model. Sections III, IV and V discuss the Permutation Monte Carlo method and the numerical instabilities in its usual implementations. In Section VI we present our proposal for avoiding this stability by rearranging terms in a key computation of the method. Section VII presents some numerical results, showing that the method is able to estimate reliability values for networks with hundreds of components. Finally, Section VIII discusses conclusions and future work.

II. 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*.

We will work with heterogeneous network models. This means that $\mathbb{P}\{X_i = 1\} = p_i$ and $\mathbb{P}\{X_i = 0\} = q_i = 1 - p_i$, $i = 1, \ldots, m$, where the p_i values may be different for the different links i.

We thus define network reliability, R, and unreliability, Q, as follows:

$$R = \mathbb{P}\{\phi(\mathbf{X}) = 1\} \quad \text{and} \quad Q = \mathbb{P}\{\phi(\mathbf{X}) = 0\} = 1 - R.$$

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 I, 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.

III. 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 [10], [23], [24], [25], [26], [27] and [28].

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 [28]. An original and efficient approach, one of whose applications can be found in [29], 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 [30] and [31].

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.

IV. Permutation Monte Carlo (PMC)

In PMC the Network Static Model is transformed into a dynamic model by the so called *Creation Process*, proposed in [10]. 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:

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

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:

$$\mathbb{P}\{X_i(1) = 1\} = \mathbb{P}\{\tau_i \leq 1\} = 1 - e^{\ln(q_i)} = 1 - q_i = r_i$$
$$\mathbb{P}\{X_i(1) = 0\} = \mathbb{P}\{\tau_i > 1\} = e^{\ln(q_i)} = q_i$$

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 $T = \{\tau_1, \tau_2, \ldots, \tau_m\}$. Calling Π to the set of all the *permutations* in T, the general form of some $\pi \in \Pi$ is:

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

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:

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

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:

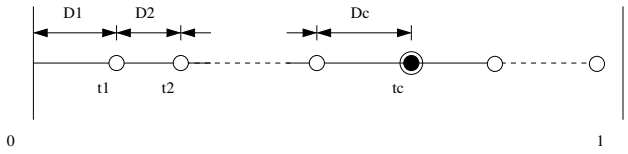

Fig. 1. The Creation Process (CP)

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

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

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

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 (1), 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:

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

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 as follows:

$$\begin{aligned}
\Delta_1 &\sim \text{Exp}(\lambda_{(1)} + \lambda_{(2)} + \ldots + \lambda_{(c)} + \ldots + \lambda_{(m)}) \\
\Delta_2 &\sim \text{Exp}(\qquad \lambda_{(2)} + \ldots + \lambda_{(c)} + \ldots + \lambda_{(m)}) \\
&\ \ \vdots \\
\Delta_c &\sim \text{Exp}(\qquad\qquad\qquad \lambda_{(c)} + \ldots + \lambda_{(m)})
\end{aligned}$$

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:

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

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:

$$\widehat{Q}_P = \frac{1}{N}\sum_{i=1}^{N}\gamma(\pi^{(i)}), \tag{2}$$

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).

It is simple to show that the variance of the estimator $\widehat{Q}_P$ in (2) is smaller than the variance of the standard estimator $\widehat{Q}_S$ in (1):

$$\begin{aligned}
\mathbb{V}\{\widehat{Q}_S\} &= q\,(1-q)\\
&= \sum_{\pi}\gamma(\pi)\,\mathbb{P}\{\Pi=\pi\}\left(1-\sum_{\pi}\gamma(\pi)\,\mathbb{P}\{\Pi=\pi\}\right)\\
&= \sum_{\pi}\gamma(\pi)\,\mathbb{P}\{\Pi=\pi\} - \left(\sum_{\pi}\gamma(\pi)\,\mathbb{P}\{\Pi=\pi\}\right)^2\\
&\quad \pm \sum_{\pi}\gamma(\pi)^2\,\mathbb{P}\{\Pi=\pi\}\\
&= \mathbb{V}\{\widehat{Q}_P\} + \underbrace{E(\gamma(\pi)) - E(\gamma(\pi)^2)}_{\geq 0}
\end{aligned}$$

what shows that $\mathbb{V}\{\widehat{Q}_P\} \leq \mathbb{V}\{\widehat{Q}_S\}$.

V. PMC Implementation issues

Once a permutation, π, is sampled, $\gamma(\pi)$ can be computed according to the distribution determined in [32], that in this context reads:

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

For $t=1$ this expression becomes:

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

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 [18] 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 [33] 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 $c = 15$:

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

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

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

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.

VI. 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:

$$\begin{aligned}
\mathbb{P}\{\Delta_1 + \ldots + \Delta_c > 1 \mid \pi\} &= \sum_{j=1}^{c} \frac{\left(\prod_{i=1}^{c} \Lambda_i\right) e^{-\Lambda_j}}{\Lambda_j \prod_{\substack{k=1 \\ k \neq j}}^{c} (\Lambda_k - \Lambda_j)} \\
&= \sum_{j=1}^{c} \frac{\Lambda_j \prod_{\substack{i=1 \\ i \neq j}}^{c} \Lambda_i}{\Lambda_j \prod_{\substack{k=1 \\ k \neq j}}^{c} (\Lambda_k - \Lambda_j)} e^{-\Lambda_j} \\
&= \sum_{j=1}^{c} \underbrace{\left(\prod_{\substack{i=1 \\ i \neq j}}^{c} \frac{\Lambda_i}{\Lambda_i - \Lambda_j}\right)}_{\alpha_j} e^{-\Lambda_j}
\end{aligned}$$

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:

$$\begin{aligned}
e^{-\Lambda_j} &= e^{-\left(\lambda_{(j)}+\ldots+\lambda_{(c)}+\ldots+\lambda_{(m)}\right)}\\
&= e^{-\lambda_{(j)}} \;\ldots\; e^{-\lambda_{(c)}} \;\ldots\; e^{-\lambda_{(m)}}\\
&= e^{-(-\ln(q_{(j)}))} \;\ldots\; e^{-(-\ln(q_{(c)}))} \;\ldots\; e^{-(-\ln(q_{(m)}))}\\
&= e^{\ln(q_{(j)})} \;\ldots\; e^{\ln(q_{(c)})} \;\ldots\; e^{\ln(q_{(m)})}\\
&= q_{(j)} \;\cdots\; q_{(c)} \;\cdots\; q_{(m)}
\end{aligned}$$

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

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

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:

$$\begin{aligned}
\sum_{j=1}^{3} \alpha_j\, e^{-\Lambda_j} &= \alpha_1\, e^{-\Lambda_1} + \alpha_2\, e^{-\Lambda_2} + \alpha_3\, e^{-\Lambda_3}\\
&= \alpha_1\, q_{(1)}q_{(2)}q_{(3)}q_{(4)}q_{(5)}q_{(6)}q_{(7)}\\
&\quad + \alpha_2\, q_{(2)}q_{(3)}q_{(4)}q_{(5)}q_{(6)}q_{(7)}\\
&\quad + \alpha_3\, q_{(3)}q_{(4)}q_{(5)}q_{(6)}q_{(7)}\\
&= q_{(3)}q_{(4)}q_{(5)}q_{(6)}q_{(7)}(\alpha_1\, q_{(1)}q_{(2)} + \alpha_2\, q_{(2)} + \alpha_3)\\
&= q_{(3)}q_{(4)}q_{(5)}q_{(6)}q_{(7)}(q_{(2)}(\alpha_1\, q_{(1)} + \alpha_2) + \alpha_3)
\end{aligned}$$

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.

VII. COMPUTATIONAL RESULTS

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

The standard Monte Carlo 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:

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

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

$$\mathbb{W}_P = \frac{t_S \times \mathbb{V}_S}{t_P \times \mathbb{V}_P}$$

where t_S and t_P are the simulation times of Standard Monte Carlo 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 studied in literature, the Dodecahedron network (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 I rows correspond to the individual link reliabilities; the columns show the unreliability estimation computed by PMC, and the speedups over the standard Monte Carlo method. It is possible to observe that in the low-reliability case, standard Monte Carlo 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.

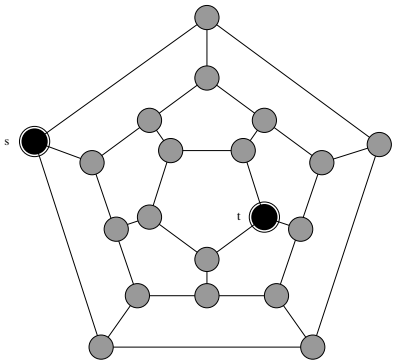

Fig. 2. Dodecahedron Network

As a second test case we present the results obtained over the G–20 network. Figure 3 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 II includes the following columns: individual link reliabilities, unreliability estimation computed by PMC, and speedup over the standard Monte Carlo 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 4) for the case of $n = 5$. The Grid–5×5 network has 25 nodes and 40 links. The results are shown in Table

TABLE I
NETWORK: DODECAHEDRON

r_i	$\widehat{Q}_P$	$\mathbb{W}_P$
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**

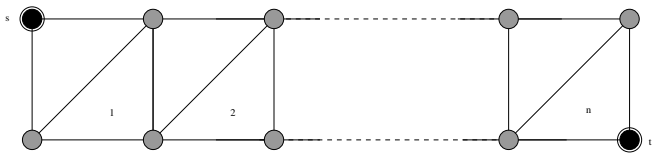

Fig. 3. G–n Networks

TABLE II
NETWORK: G–20

r_i	$\widehat{Q}_P$	$\mathbb{W}_P$
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**

III, including again the same columns: individual link reliabilities, unreliability estimation computed by PMC, and speedup over the standard Monte Carlo method. Five different experiments are reported for each topology, corresponding to the link reliability values $r_i = 0.9; 0.99; 0.999; 0.9999; 0.99999$.

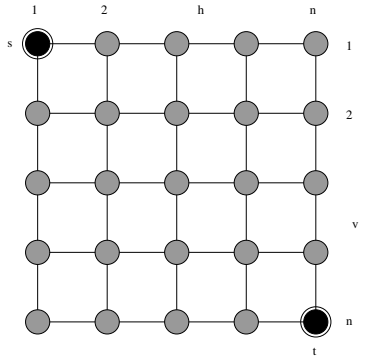

Fig. 4. Grid–$n \times n$ Network

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 Monte Carlo; without being hampered by the small numbers/potential underflow effect of the

TABLE III
NETWORK: GRID–5×5

r_i	$\widehat{Q}_P$	$\mathbb{W}_P$
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**

original PMC proposal. The results are consistently good for networks of different sizes.

Now we show, in Table IV, 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 $n = 8$, $n = 9$ and $n = 10$ have, respectively, 112, 144 and 180 links.

TABLE IV
$\widehat{Q}_P$ FOR 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

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 I, II and III were, respectively, 10×10^6, 1×10^6 and 4×10^6, whereas, the number of replications for the Standard Monte Carlo simulations were 100×10^6, 20×10^6 and 50×10^6. These values were chosen so that all the individual simulations would run for approximately 100 seconds.

In the case of Table IV, 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 Monte Carlo; 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.

VIII. 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 new 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.

Future work can include studying other simulation methods from literature and determine which are the boundaries in network size and link reliability values which guarantee robust results. 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] 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.

[6] 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.

[7] 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.

[8] 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.

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

[10] 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.

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

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

[13] 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.

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

[15] 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.

[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] 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.

[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] 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.

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

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

[23] 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

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

[25] 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

[26] 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.

[27] 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

[28] 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.

[29] 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.

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

[31] 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

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

[33] 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