International Journal of Performability Engineering Vol. 10, No. 2, March 2014, pp. 163-172.

Printed in India

Network Reliability Monte Carlo With Nodes Subject to Failure

ILYA GERTSBAKH[1], YOSEPH SHPUNGIN[2], R. VAISMAN[3]
[1]*Department of Mathematics, Ben-Gurion University*
P. O. Box 653, Beer-Sheva, 84105, ISRAEL
[2]*Software Engineering Department*
Sami Shamoon College of Engineering, Beer Sheva 84100 ISRAEL
[3]*Faculty of Industrial Engineering and Management*
Technion, Israel Institute of Technology, Haifa, ISRAEL

(Received on June 04, 2013, revised on November 15, 2013)

Abstract: We extend the network reliability estimation methodology based on evolution (creation) Monte Carlo into four directions: (i) introducing unreliable nodes; (ii) adjusting the evolution process with merging to "closure" operation suitable for unreliable nodes; (iii) in case of numerical instability in computing convolutions, we suggest a special Monte Carlo algorithm based on importance sampling; (iv) we extend the traditional network terminal connectivity criterion to criteria describing network disintegration into a critical number of clusters, or the critical size of the largest component.

Keywords*: evolution Monte Carlo, unreliable nodes, closure for nodes, critical number of clusters, simulating convolutions , numerical instability.*

1. Introduction

Abbreviations and Notation:

CMC- crude Monte Carlo;
EMC -evolution Monte Carlo;
Q - network failure (*DOWN*) probability;
RE -relative error;
RTV -relative time variance;
CPU-central processor unit;
N - number of simulation runs;
$\tau_i \sim Exp(\Lambda_i)$ - random variable τ_i has exponential distribution with parameter Λ_i

The traditional network reliability problem, as it has been stated in the recently published "Handbook of Monte Carlo Methods" [8] is formulated as follows. Let $G(V, E, K)$ be an undirected graph (network) with V being the set of n nodes (or vertices), E being the set of m edges (or links), and $K \subseteq V$, $|K| = s$, being a set of special nodes called *terminals*. Associated with each edge $e \in E$ is a Bernoulli random variable $X(e)$ such that $X(e) = 1$ corresponds to the event that the edge is operational (*up*) and $X(e) = 0$ corresponds to the event that the edge has failed (is *down).* The edge failures are assumed to be independent events. Assume that $P(X(e) = 1) = p(e)$.

Based on this model, the network reliability $R = P(UP)$ and unreliability $Q = P(DOWN) = 1 - R$ is defined as the probability that a set of terminal nodes K is connected (not connected) in the sub-graph containing all of the nodes of V. When $s = 2$, the model describes so-called $s - t$ terminal connectivity. When $s = n$, we have all-node connectivity.

This model, although very simple, has been employed in a wide number of application

*Corresponding author's email: elyager bezeqint.net

settings to communication networks, mobile ad hoc and tactical radio networks, evaluation of transport and road networks, see [1, 3, 4, 7, 9, 10].

One of the most computationally efficient methods for calculating network reliability for the above model is based on so-called *evolution* or *creation* process first suggested in [2]. It works as follows. Initially, at $t = 0$ all edges e are *down*. At some random moment $\xi(e)$, edge e is "born", independently of other edges, and remains in state *up* forever. $\xi(e)$ is assumed to be exponentially distributed with parameter $\lambda(e)$:

$$P(\xi(e) \le t) = 1 - e^{-\lambda(e)t}, e \in E.$$

Fix an arbitrary moment $t = t_0$, for example, $t_0 = 1$. Choose for each e its "birth rate" $\lambda(e)$ so that the following condition holds:

$$P(\xi(e) > t_0) = e^{-\lambda(e)t_0} = 1 - p(e).$$

This formula means that at time t_0 the edge e has already born (is *up*) with probability $p(e)$. The crucial observation is that the snap-shot of the whole network taken at t_0 gives the picture of the whole network which coincides in probability with its state produced as a result of generating the state of each edge with static probability $p(e)$ of being *up* and $1 - p(e)$ of being *down*.

The Monte Carlo procedure for estimating network *UP* probability R is implemented by generating so-called *trajectories* which imitate the development in time of the evolution process. Let us consider an example, see Fig. 1.

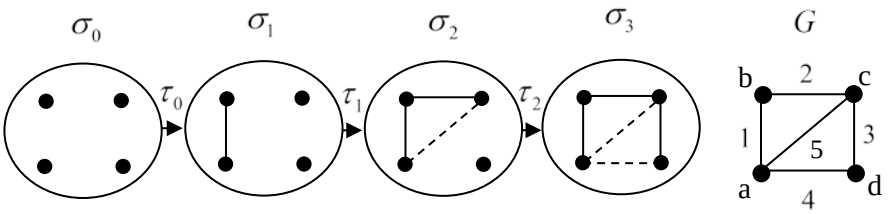

Figure 1: Evolution Process. Edges are born in the sequence: $1 \rightarrow 2 \rightarrow 3$.

We have a four node network with five edges. The network is *UP* if all its nodes are connected to each other. The initial state without edges at $t = 0$ is denoted as σ_0. The network stays in it during random time τ_0 which is exponentially distributed with parameter $\Lambda_0 = \sum_{i=1}^{5} \lambda_i$. Suppose the edge 1 is born first. By the properties of exponential distribution, this happens with probability λ_1/Λ_0. Then the system goes into its next state σ_1. Now in this state the system spends random exponentially distributed time $\tau_1 \sim Exp(\Lambda_1 = \sum_{i=2}^{5} \lambda_i)$. Suppose that the next edge born is 2.

This happens with probability $\lambda_2/(\lambda_2 + \ldots + \lambda_5)$. This transfers the system into state σ_2. Now note that at this stage of the evolution process we *can add edge* 5 (shown by dotted line) to already born edges and *exclude* it from the further evolution process because the existence or nonexistence of this edge does not affect the already formed component of three nodes a, b, c created by edges 1 and 2, see closure or merging operation in [2,8]. The system spends in σ_2 random time $\tau_2 \sim Exp(\lambda_3 + \lambda_4)$. Suppose edge 3 is born first, which happens with probability $\lambda_3/(\lambda_3 + \lambda_4)$. Then the system enters the state σ_3 which is, by definition, the network *UP* state. Note that the random times τ_0, τ_1, τ_2 are independent, and the trajectory $\omega = \{\sigma_1 \rightarrow \sigma_2 \rightarrow \sigma_3\}$ takes place with probability

$$P(\omega) = \frac{\lambda_1}{\Lambda_0} \cdot \frac{\lambda_2}{\Lambda_1} \cdot \frac{\lambda_3}{\Lambda_2}.$$

Finally, let us find out the probability $P(UP; \omega)$ that the network will be *UP* given that the evolution goes along this trajectory:

$$P(UP; \omega) = P(\tau_0 + \tau_1 + \tau_2 \le t_0; \omega).$$

This probability can be found in a closed form using well-known hypo-exponential distribution, see [11], page 299 or [3], Appendix B. It is worth to present the corresponding formula in its general form.

Let independent random variables $\tau_i \sim Exp(\Lambda_i)$, $i = 0,1,2,\ldots,r-1$. Suppose that

$$\Lambda_0 > \Lambda_1 > \Lambda_2 > \cdots \Lambda_{r-1}.$$

Then

$$P(\sum_{i=0}^{r-1} \tau_i \le t_0) = 1 - \sum_{i=0}^{r-1} e^{-\Lambda_i t_0} \prod_{j \ne i} \frac{\Lambda_j}{\Lambda_j - \Lambda_i}. \quad (1)$$

Now suppose that we have generated M trajectories $\omega_1, \ldots, \omega_M$ and for each ω_i we have calculated by (1) the corresponding convolution $P(UP; \omega_i)$. The unbiased estimate of network *UP* probability is found as an average

$$\hat{P}(UP) = \frac{\sum_{i=1}^{M} P\,(UP; \omega_i)}{M}. \quad (2)$$

In the next section we show how an analogous process can be implemented for the case of unreliable nodes.

2. Evolution Process for Networks with Unreliable Nodes

2.1 How it works?

Similarly to Introduction, we are dealing with an undirected graph (network) $G(V, E, K)$. V is the set of n nodes (or vertices), E is the set of m edges (or links), and $K \subseteq V$, $|K| = s$, is the set of special nodes called *terminals*. Contrary to Introduction, we now assume that the edges are always *up*, and the nodes, except the terminal nodes, are subject to failures and fail independently. Terminal nodes are assumed to be always operational (*up*). So, there are $n_1 = n - s$ nodes subject to failure. Let us denote the nodes $\alpha_1, \alpha_2, \ldots, \alpha_{n_1}$. Node α_i is *down* with probability q_i and *up* with probability $p_i = 1 - q_i$. Node failure means that all edges incident to this node are erased and the failed node becomes isolated. If nodes α_i and α_j both are *up* and there exists in E an edge $e(i,j)$ connecting these nodes, then edge $e(i,j)$ becomes operational (*up*). If one of two nodes that are connected in G by an edge is *down*, then the corresponding edge is assumed to be nonexisting (erased).
Networks with unreliable nodes have wide range of applications, especially in models of network survivability, see *e.g.*, [3, 4, 6].

Our main tool for investigating network reliability will be evolution (creation) process similar to the described in Introduction.

Associate with each node α_i a random variable $\xi_i \sim Exp(\lambda_i)$, assume that all ξ_i are independent, and define λ_i for an arbitrary $t_0 > 0$ in such a way that it satisfies the following equality:

$$\mathrm{P}(\xi_i > t_0) = \mathrm{e}^{-\lambda_i t_0} = 1 - p_i. \quad (3)$$

At $t = 0$ all nodes are *down* (except the terminal nodes). Put λ_i to be the birth rate of α_i. After a node is "born", it remains *up* forever. At $t = 0$, also no edges exist. If in E there is an edge $e(i,j)$ connecting nodes α_i and α_j, and both they are born, then immediately the

edge $e(i,j)$ "comes to life". The principal fact is that the snap-shot of the network taken at $t = t_0$ will show that the network is *UP* with the probability equal to $P(UP)$ calculated for the original static model. Now, as in the Introduction, we construct the trajectories of the evolution process leading from the initial state (no nodes born and no edges) to the network *UP* state which guarantees the terminal connectivity.

Let us consider an example. Figure 2 shows a ladder-type network with terminal nodes s, t (bold). It has 8 nonterminal nodes and 12 edges. Nodes $i = 1,2,\ldots,8$ fail with probability q_i. Network state at $t = 0$ is σ_0, with no nodes and no edges. Nodes which are not born are shown by empty circles. Suppose node 4 is born the first. This leads to the network state σ_1. The born node is shown by bold circle. Automatically, edge $e(t,4)$ appears, and we see that there are two paths leading from 4 to t: one is direct $4 \to t$ and the second is $4 \to 2 \to 1 \to t$ which is "parallel" to it. Obviously, nodes 1 and 2 are nonrelevant since every connection of node 4 to node t via nodes 1,2 is blocked by the direct edge $(4,t)$. So, nodes 1 and 2 constitute a "closure" and are declared as being *up*. We mark them by non bold circles. Suppose, the next birth belongs to node 5, see the state σ_2. In a similar way, nodes 7 and 8 belong to closure and are declared as being *up*. Let the next birth belongs to node 3. It leads to appearance of edges $e(t,3)$ and $e(3,5)$ and therefore leads to the network *UP* state because there exist a path $s \to 5 \to 3 \to t$.

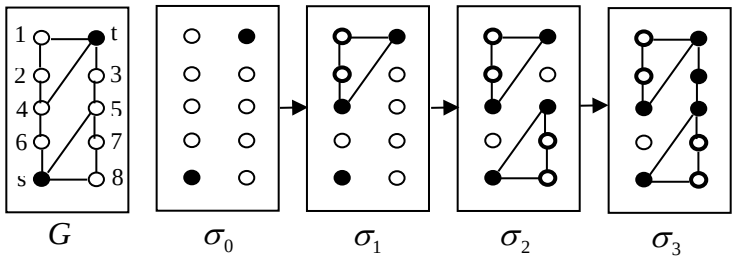

Figure 2: Nodes are born in the following order: $4 \to 5 \to 3$. Nodes 1, 2 and 7, 8 are added as closure.

The trajectory of births leading from σ_0 to *UP* is therefore $\omega = \{4 \to 5 \to 3\}$. Let us compute the probability $P(\omega)$ of this trajectory. Following the reasoning provided in Introduction, we find out that

$$P(\omega) = \frac{\lambda_4}{\sum_{i=1}^{8} \lambda_i} \cdot \frac{\lambda_5}{\sum_{i \neq 1,2,4} \lambda_i} \cdot \frac{\lambda_3}{\lambda_6 + \lambda_3}.$$

The sojourn time in σ_0 is $\tau_0 \sim Exp(\sum_{i=1}^{8} \lambda_i)$. Sojourn time in σ_1 is $\tau_1 \sim Exp(\sum_{i \neq 1,2,4} \lambda_i)$, and in state σ_2 the system stays random time $\tau_2 \sim Exp(\lambda_6 + \lambda_3)$. Thus the snapshot of the network taken at time t_0 sees the network as being *UP* with probability $P(\tau_0 + \tau_1 + \tau_2 \leq t_0; \omega)$.#

In general, suppose that on certain stage of the evolution process (after carrying closure operations) we arrived at state σ^*. Denote by Λ^* the sum of birth rates of all nodes which are not born yet on this stage. Denote the set of all these "active" nodes by A^*. So, $\Lambda^* = \sum_{\alpha_i \in A^*} \lambda_i$. The next node α_j will born with probability

$$\rho^* = \frac{\lambda_j}{\Lambda^*}.$$

After this birth takes place, we arrive (after carrying out the closure, if possible, and

adding edges between already born nodes) at the state σ^{**} which is the direct successor of state σ^{*}. In general, we will simulate a trajectory ω of states which starts from trivial state σ_0 and ends in some state $\sigma_x \equiv UP$:

$$\omega = \{\sigma_0 \to \sigma_1 \to \sigma_2 \to \cdots \to \sigma_x \equiv UP\}.$$

Let $\tau_i \sim Exp(\Lambda_i)$ be the sojourn time in state σ_i, $i = 0,1, \ldots, x-1$.
Denote

$$R(\omega) = P(UP;\omega) = P(\sum_{i=0}^{x-1} \tau_i \le t_0).$$

2.2 The Simulation Algorithm

Now we are able to describe the simulation algorithm which implements the above described procedure.

Algorithm 1.

1. Put $\hat{R} := 0$
2. Generate trajectory ω leading from the trivial state σ_0 to *UP* state of the network. Use for its generation the above described transition probability from state σ^{*} to its successor $\sigma **$. After a transition takes place, locate non relevant nodes and declare them as being *up*. Add edges between the *up* nodes if such edges are present in the set E.
3. Using (1), calculate $R(\omega)$ and put $\hat{R} := \hat{R} + R(\omega)$.
4. Repeat **2** and **3** M times.
5. Put $\hat{R}(UP) := \frac{\hat{R}}{M}$.

Obviously, $\hat{R}(UP)$ is an unbiased estimate of network *UP* probability $R = P(UP)$.

An important remark: node α_i birth probability is defined as $\lambda(\alpha_i) = -\frac{\log[q(\alpha_i)]}{t_0}$, or $q(\alpha_i) = e^{-\lambda(\alpha_i)t_0}$. Therefore the transition probabilities in the evolution process from state to state do not depend on t_0. If we increase t_0, we will correspondingly decrease the node failure probabilities, but this will not affect the probabilities of generating the ω trajectories. Therefore, the rare event phenomenon will not take place for this algorithm.

2.3 More about Closure

Let us describe in more formal way the node closure operation. Suppose that after some nodes were born, we have network state σ. Denote by S_r the set of nodes which are already born. Let us call them "red" nodes. For simplicity we will assume that the terminal nodes also are "red". Denote the remaining set of nodes $S_w = V \setminus S_r$ and call the nodes in this set "white" nodes. Let the edges between white nodes also be called white; let the edges between red edges be called red, and let the edges between nodes of different color be called “green”. Consider a particular stage of the birth process, say σ_2 on Fig. 2. Edges $e(1,2)$ and $e(7,8)$ are white, edges $e(4,t)$ and $e(s,5)$ are red, and edges $e(2,4), (1,t), e(s,8), e(5,7)$ are green. The set of red nodes connected by red edges consists of several components. So, for σ_2 there are two such red components: $\{4,t\}$ and $\{s,5\}$. The nodes in these components are connected by red edges only. Similarly, the set of white nodes also consists of several components, namely $\{1,2\}$ and $\{7,8\}$.

The general rule for carrying out the closure is the following:
Let G_w be a white component which is connected by green edges to *only one* red

component G_r. Then all nodes of G_w can be declared red and joined to G_r.
For example, $G_w = \{1,2\}$ can be declared red and joined to $G_r = \{4, t\}$.

The implementation of the closure for unreliable nodes is algorithmically considerably more involved than for unreliable edges because it amounts to search for all white components on each stage of the evolution process. The CPU time spent on this search may exceed the gain obtained by reducing the length of the simulated trajectories. In the next section we will show an example comparing the simulation algorithms with and without closure.

2.4 Fighting Numerical Instability for Long Convolutions

Our numerical experience of simulating relatively large networks (e.g. having more than 40-50 nodes) reveals that the use of (1) may show a *numerical instability* resulting in getting absurd results, like negative probability or probability exceeding 1. The reason is that for long trajectories, the Λ_i values may become very large, as a result of which the terms in (1) may become very large, especially when some Λ's happen to be close to each other. The terms in (1) have *alternating* signs and the computer calculations according to (1) may become extremely unstable. We met this phenomenon already in preparing numerical examples for our paper [2]. Also some Monte Carlo researchers have reported to us about the same instability in private communications. To avoid it, we suggested in [5] an alternative to (1) based on importance Monte Carlo sampling procedure. Detailed description and the corresponding proofs can be found also in [3], Chapter 7.
Here is a short description of the corresponding algorithm. Suppose we want to estimate

$$P(\tau_1 + \tau_2 + \ldots + \tau_m \le T),$$

where τ_i are independent nonnegative random variables (r.v.'s) with density $f_i(t)$ and cumulative distribution function $F_i(t)$.

Algorithm 2

1. Simulate r.v. X_1 with the density $f_1(t)/F_1(T)$. (This r.v. has support $[0,T]$). Let $X_1 = x_1$.
   Simulate r.v. X_2 with density $f_2(t)/(F_2(T - x_1))$. (This r.v. has support $[0, T - x_1]$). Let $X_2 = x_2$. Continue recursively and generate r.v. X_k having density $f_k(t)/(F_k(T - x_1 - x_2 - \cdots - x_{k-1}))$.
2. After r.v.'s $X_1, X_2, \ldots X_{m-1}$ have been generated, calculate
   $B_m(T) = F_1(T) \cdot F_1(T - x_1) \cdot F_3(T - x_1 - x_2) \cdot \ldots \cdot F_m(T - x_1 - x_2 - \cdots x_{m-1})$.
3. Repeat steps **1, 2** K times and calculate $\hat{B}(T) = \dfrac{\sum_{i=1}^{k} B_m^i(T)}{K}$.

It was proved in [3,5] that $\hat{B}(T)$ is an *unbiased estimate* of $P(\tau_1 + \tau_2 + \ldots + \tau_m \le T)$. The best number of replications K must be found experimentally. Small values of K have smaller CPU time but have greater relative error. We demonstrate the choice of K in our example 4 in the next section.

2.5 Modification of Network UP- Definition

In studying models dealing with network survivability subject to random attack on its nodes, the researchers are interested in evaluating the network ability to retain its functionality when a certain part of nodes becomes damaged (i.e., becomes *down*). Very often the network stops functioning if it disintegrates into too many isolated components. Of particular interest is the following criterion. Suppose that the network has s special

nodes representing important facilities (hospitals, storages, communication centers, etc). Let us call them *vital nodes* (VN's). Assume that the VN's do not fail. The network, by definition, is functional (*UP*) if and only if in the process of node failures it falls apart into no more than r so-called clusters, $r \leq s$. A cluster is a (connected) component containing at least one VN. So, for example, the network represents a railroad system with $s = 5$ VN's. Initially, all VN's constitute one cluster. The network remains operational if and only if it has less than four clusters, i.e. has one, two or three clusters. (Here $r = 3$).

An important feature is that the above described evolution process built on network nodes may be used for calculating network *UP* probability also for this modified criterion. We simply simulate the node birth trajectory and end it if the number of clusters becomes less or equal r. An example of using this criterion will be given in the next section.

3. Simulation Examples

Before we present the simulation results, let us make the following principal comment. Suppose we determine all $\lambda(e)$ values for some fixed $t_0 = 1$. What will happen if we take the snap-shot of the system at another instant, say $t_1 = 0.9$? This is equivalent to multiplying all lambda values by the factor $t_0/t_1 = 1/0.9 = 1.11$. This operation will *not change* the probabilistic mechanism of creating the trajectories of the evolution process because the transitions probabilities depend on the ratios of the lambda values. Therefore, introducing another t_0 can be made by carrying out only one extra arithmetic operation - computing the convolution by (1) for different t_0 value. This will demand a negligible amount of extra CPU time. Therefore, when we compare the efficiency of CMC and EMC we will take into account that the EMC allows to compute network reliability for *several* sets of node unreliability values $q_1, \ldots, q_m$ using the same CPU time. We make an agreement that the relevant CPU time presented below is given for *five* different sets of q values. In literature, it is suggested to measure the efficiency of Monte Carlo procedure by RTV which equals the CPU time times the squared relative error. The factor of 0.2 will be introduced in all RTV values presented below. We will present the following

Example 1: Terminal Connectivity

We consider a 9x9 rectangular grid with five terminals located in the center and in the corners, see Fig. 3. Table 1 compares the simulation results of CMC and EMC, for two sets of node failure values. The first set is $q_1 = 0.001, q_2 = 0.01$ and the second set is $q_1 = 0.001, q_2 = 0.001$, where q_1 corresponds to odd and q_2 to even node, respectively. (Node is odd if sum of its coordinates is odd and even, otherwise. For example, the central node is even, two closest neighbors of a corner are odd). There are in total 76 nodes subject to failure.

The first two rows of the Table 1 compare the CMC and EMC for all-terminal connectivity. The last two rows compare the CMC and EMC for the $s - t$ connectivity when s and t are located on the opposite ends of the diagonal. The EMC was implemented without applying the closure.

Let us make an important observation related to the lower half of Table 1 ($q = 0.001$ for all nodes), in rows three and four. Network fails if one of two terminals gets isolated. This happens if two neighboring nodes fail, the probability of which is $q^2 = 10^{-6}$. There are two such min cuts, and therefore the probability of the main event leading to the loss of $s - t$ connectivity is about $2 \cdot 10^{-6}$. This is in excellent agreement with the results presented in the last two lines of the table. The above reasoning shows that the Burtin-

Pittel approximation (see [3], p.70,73) is very accurate.

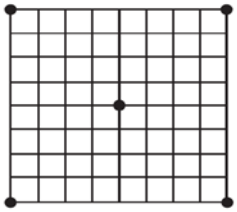

Figure 3: 9x9 Grid with Five Terminals (bold)

Table 1: The Simulation Results for Terminal Connectivity Criterion

Method	Q	RE	RTV	CPU	N	q_1, q_2
CMC	$1.88 \cdot 10^{-5}$	0.24	1.14	19.9	1,000,000	0.001;0.01
EMC	$1.97 \cdot 10^{-5}$	0.086	$0.17 \cdot 0.2 = \mathbf{0.034}$	22.0	100,000	0.001;0.01
CMC	$2.01 \cdot 10^{-6}$	0.23	9.4	175	1,000,000	0.001;0.001
EMC	$2.04 \cdot 10^{-6}$	0.086	$0.036 \cdot 0.2 = \mathbf{0.0072}$	187	1,000,000	0.001;0.001

Example 2: Network Disintegration into Several Clusters

Here we consider network survivability with respect to the number of clusters. The data presented in Table 2 are for the 9x9 grid with 5 terminals shown on Fig. 3. The first two rows present data when the network is *UP* if it has 1 or 2 clusters. The next two rows present similar data when the network is *UP* if it has 1,2 or 3 clusters; the last two rows present data for *UP* defined as having 1,2,3 or 4 clusters. We will denote the above *UP* states as *UP*(1,2), *UP*(1,2,3) and *UP*(1,2,3,4). *DOWN* state in the last case means disintegration into 5 clusters, i.e. complete isolation of all terminals. $Q = 1 - P(UP)$. As in the previous example, we use EMC without closure.

Table 2: The Simulation Results for Disintegration into Several Clusters

Method	Q	RE	RTV	CPU	N	q_1, q_2
CMC	$3.08 \cdot 10^{-4}$	0.081	0.063	97	500,000	0.05/0.1
EMC	$3.04 \cdot 10^{-4}$	0.035	$0.15 \cdot 0.2 = \mathbf{0.03}$	121	500,000	0.05/0.1
CMC	$1.46 \cdot 10^{-6}$	0.38	14.5	98	5,000,000	0.05/0.1
EMC	$1.25 \cdot 10^{-6}$	0.19	$4.07 \cdot 0.2 = \mathbf{0.81}$	103	500,000	0.05/0.1
CMC	—	—	—	—	—	—
EMC	$1.67 \cdot 10^{-9}$	0.38	$13.8 \cdot 0.2 = \mathbf{2.8}$	90	500,000	0.05/0.1

Remark 1. The network is highly reliable with respect to disintegration into 5 clusters, see the last row. For this case, CMC was not used since it would demand $N = 10^{10}$ experiments and CPU time of several hours#.

Example 3: Monte Carlo with Closure Operation for Nodes.

We carried out the experimentation on a random Erdos-Renyi graph with 30 nodes and 48 edges, for $s - t$ terminal connectivity, see Fig. 4.

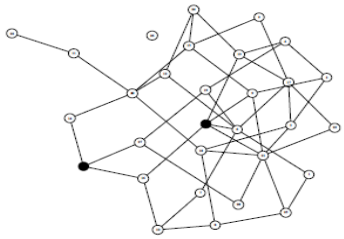

Figure 4: Replica of a Random Graph with 30 Nodes and 48 Edges

We compared three methods: the CMC, the EMC and the EMC with closure operation. It was found experimentally that the closure starts working after approximately one third of the nodes (i.e. about 16 nodes) are already born. Table 3 presents the simulation results. What follows from this table is that the use of closure for unreliable nodes gives a small gain in RE (because the trajectories become shorter) but demands considerably more CPU time. In total, there is no gain in RTV. As we noticed before, the closure for unreliable *edges* needs only finding out the edges whose both ends belong to the already existing component. As to the nodes, the search for non-relevant nodes needs searching for connected groups of nodes which can be united with one of the already existing component. As to the nodes, the search for non-relevant nodes needs searching for connected groups of nodes which can be united with one of the already existing components of born nodes. Our conclusion is that from practical point of view, there is no need to complicate the EMC algorithm with the closure operation.

Table 3: The Simulation Results for Erdos-Renyi Graph

Method	Q	RE	RTV	CPU	N	q
CMC	$2.2 \cdot 10^{-6}$	0.314	3.25	32.1	5,000,000	0.1
EMC	$2.03 \cdot 10^{-6}$	0.047	$0.35 \cdot 0.2 = \mathbf{0.07}$	15.8	500,000	0.1
EMC & closure	$2.01 \cdot 10^{-6}$	0.029	$0.040 \cdot 0.2 = \mathbf{0.08}$	46.6	500,000	0.1

Example 4: Loss of Stability and Simulating Convolutions.

Loss of stability usually takes places when the trajectories are long. To present an example of high reliable graph, we simulated $s - t$ connectivity in a random graph with 100 nodes and 1000 edges. The reliability criterion was the $s - t$ connectivity between two randomly chosen terminal nodes. The odd/even nodes have low reliability, $q = 0.5$ and $q = 0.6$. High reliability is achieved because the graph is quite dense. The average length of a trajectory is about 7 nodes. Table 4 presents the simulation results for CMC and various K. It follows that the best results for RTV are obtained for $K = 20$.

Table 4: The Simulation Results for Highly Reliable Random Graph

Method	Q	RE	RTV	CPU	N	$q_1; q_2$	K
CMC	$2.4 \cdot 10^{-6}$	0.74	24.6	45.0	1,000,000	0.5;0.6	—
Sim. conv.	$2.82 \cdot 10^{-6}$	0.254	$5.08 \cdot 0.2 = \mathbf{1.02}$	78.8	1,000,000	0.5;0.6	10
Sim. conv.	$2.83 \cdot 10^{-6}$	0.221	$4.07 \cdot 0.2 = \mathbf{0.81}$	82.9	1,000,000	0.5;0.6	**20**
Sim. conv.	$2.57 \cdot 10^{-6}$	0.208	$4.62 \cdot 0.2 = \mathbf{0.92}$	105.3	1,000,000	0.5;0.6	40

4. Conclusions

The Evolution Monte Carlo (EMC) as presented in the paper is a good and reliable working tool for the reliability investigation of networks with independently failing nodes and considerably superior to CMC in case of highly reliable networks (say with failure probability $Q < 10^{-5}$). From practical point of view, there is no need to incorporate into the standard EMC algorithm the analogue of closure operation. This operation gives little savings on the length of the evolution trajectories but at the same time considerably increases the CPU time for the search of all non-relevant nodes.

The general framework of the EMC algorithm allows to include new extensions of the traditional network reliability definition as K-terminal reliability. Examples of these extensions are disintegration of the network into a critical number of clusters or network failure when its maximal component contains less than $L_{\min}$ nodes. However, in certain situations, use of convolution formula (1) may display loss of numerical stability which leads, in turn, to obtaining absurd results, like having negative probabilities. If this phenomenon takes place, it is advisable to use the Algorithm 2 described in Section 2.4.

References

[1] Cook, J. L. and J. I. Ramirez-Marquez. *Two-Terminal Reliability Analysis for a Mobile Ad Hoc Wireless Network*. Reliability Engineering & System Safety, 2007; 92(6); 572-581.

[2] Elperin, T., I. B. Gertsbakh, and M. Lomonosov. *Estimation of Network Reliability using Graph Evolution Models.* IEEE Transactions on Reliability, 1991; 40(5); 572–581.

[3] Gertsbakh, I., and Y. Shpungin. *Models of Network Reliability: Analysis, Combinatorics and Monte Carlo*. CRC Press, 2009.

[4] Gertsbakh, I., and Y. Shpungin. *Network Reliability and Resilience*, Springer Briefs in Electrical and Computer Engineering, Springer, 2011.

[5] Gertsbakh, I., and Y. Shpungin. *Product-Type Estimator of Convolutions.* In Semi Markov Models and Applications, J. Janssen and N. Limnios Eds). Kluwer Academic Publishers, 1999; 201-206.

[6] Gertsbakh,I., and Y. Shpungin. *Stochastic Models of Network Survivability*. Quality Technology and Quantitative Management, 2012; 9(1); 45-58.

[7] Gunnec, D., and F. S. Salman. *Assessing for the Reliability and Expected Performance of Network under Disaster Risk. OR Spectrum*, 2011; 33(3), 499-523.

[8] Kroese, D., T. Taimre, and Z. I. Botev. *Handbook of Monte Carlo Methods*, Wiley, New York, 2011; 567–574.

[9] Marseguerra, M., E. Zio, L. Podofillini, and D. W. Coit. *Optimal Design of Reliable Network Systems in Presence of Uncertainty*. IEEE Transactions on Reliability, 2005; 54(2); 243-253.

[10] Murray, L., H. Cancela, and G. Rubino. *A Splitting Algorithm for Network Reliability Estimation.* IEE Transactions, 2012; 45; 177-189.

[11] Ross, Sheldon M. *Introduction to Probability Models*, 9th Edition, Academic Press, 2007.

Short biographies of **Ilya Gertsbakh and Yoseph Shpungin** are published on page 378 of IJPE, Vol. 8, No. 4, July 2012 in their paper on *Signatures and D-spectra and their Use in Reliability Theory.*

Radislav Vaisman is a post-doctoral research fellow in the University of Queensland, Brisbane, Australia. He received his Ph.D. in Information Systems Engineering and Operational Research from Technion, Israel. He is a coauthor of one book and has several publications in international scientific journals. His field of research includes rare event simulation, randomized algorithms and network reliability.