Improved Sampling Plans for Estimating Terminal Network Reliability

Radislav Vaisman, Dirk P. Kroese, Ilya B. Gertsbakh

***Abstract*—Terminal network reliability problems appear in many real–life applications, such as transportation grids, social and computer networks, communication systems, etc. For most variants of the terminal reliability problem no analytical solution is known and one has to rely on different approximation techniques. An example of such a method is Permutation Monte Carlo, which is thought to provide fast and reliable estimates of the network reliability. In this paper we show that this simple and widely followed sampling plan is not sufficient for adequate estimation when dealing with highly reliable networks. As an alternative we propose a different sampling strategy that is based on the recently pioneered Stochastic Enumeration algorithm for tree cost estimation. We show that thanks to its built–in splitting mechanism this method is able to deliver accurate results while employing a relatively modest sample size. Moreover, our numerical results indicate that the proposed sampling scheme is capable of solving problems that are far beyond the reach of the simple Permutation Monte Carlo approach.**

***Index Terms*—Network Reliability, Permutation Monte Carlo, Stochastic Enumeration, Rare Events, Splitting.**

I. Introduction

NETWORKS have become an important part of our daily activities and as a consequence a natural question of their reliability comes to light. The network reliability problem can be stated as follows [1]. Suppose we are given an undirected graph $G(V, E, K)$, where V and E are the vertex and edge sets respectively and $K \subseteq V$ is a set of "terminal" nodes. We assume that the vertices are absolutely reliable (that is, they never fail) but that the edges are subject to failure. For $e \in E$ let the corresponding failure probability be q_e; that is, e can be in an *up* or *down* state with probabilities $p_e = 1 - q_e$ and q_e respectively. Under this setting we can ask for the probability that the terminal set K is connected (network *UP* state) or disconnected (network *DOWN* state) [2].

As was noted by Karger [3], the terminal network reliability problem belongs to the #P–complete complexity class. Consequently, it is hard (or even impossible) to solve exactly in polynomial time. The #P complexity class, introduced by Valiant [4] consists of the set of *counting* problems that are associated with a decision problem in NP (non–deterministic polynomial time), e.g., how many solutions does a propositional formula have (#SAT). The #P–complete complexity class is a sub–class of #P consisting of those problems in #P to which any other problem in #P can be reduced via a polynomial reduction. #SAT, for example, is #P–complete. Interestingly, various #P–complete problems are associated with an easy decision problem, i.e., the corresponding decision problem is in P (polynomial time), such as satisfiability of propositional formulas in *Disjunctive Normal Form* (DNF).

For some #P-complete problems there are known efficient approximations. For example, Karp and Lubby [5] introduced a *fully polynomial randomized approximation scheme* (FPRAS) for counting the solutions of DNF satisfiability formulas. Similar results were obtained for the knapsack and permanent counting problems by Dyer and Jerrum et al. [6], [7]. Moreover, Karger [3] obtained a FRPAS for all–terminal network reliability. Unfortunately, there are also many negative results. For example, Dyer et al. and Vadhan [8], [9] showed that counting the number of vertex covers remains hard, even when restricted to planar bipartite graphs of bounded degree or regular graphs of constant degree.

There are two main approaches for tackling such difficult counting problems. The first is *Markov Chain Monte Carlo* (MCMC) and the second is *Sequential Importance Sampling* (SIS). Both approaches exploit the finding of Jerrum et al. [10] that counting is equivalent to uniform sampling over a suitably restricted set. MCMC methods sample from such restricted regions by constructing an ergodic Markov chain with stationary and limiting distribution equal to the desired uniform distribution. A number of MCMC approaches with good empirical performance have been proposed; see [11], [12], [13], [14]. Botev et. al. [15] applied a generalized splitting algorithm [11] to handle the terminal network reliability problem.

There are also many examples of successful SIS implementations on various counting problems; see, for example, [5], [16], [17], [18]. More recent advances and background material can be found in [19]. Unsurprisingly, SIS was also applied in context of network reliability, see [20] for details.

In this paper we focus on the estimation of the so–called *Spectra* or *Network Signature* [2], [21], [22], [23], [24], [25] measure of a reliability network. The latter is a well–known quantity in network reliability. As soon as the Spectra is available, one can readily calculate the network reliability. Unfortunately, the Spectra is rarely available analytically, hence it is usually estimated via *Permutation Monte Carlo* (PMC). The latter is believed to provide reliable estimations for network reliability [22]. Nevertheless, we show that this widely used PMC algorithm fails to provide a reliable estimates of the Spectra when dealing with highly reliable networks, underestimating (in orders of magnitude) the true quantity of interest. The reason for this inaccuracy lies in the importance of small (rare event) probabilities that cannot be properly estimated via

Radislav Vaisman and Dirk P. Kroese are with the School of Mathematics and Physics, The University of Queensland, Brisbane 4072, Australia, e-mail: r.vaisman@uq.edu.au, kroese@maths.uq.edu.au. Their research was supported by the Australian Research Council under grant number CE140100049.

Ilya B. Gertsbakh is with Department of Mathematics, Ben Gurion University, Beer-Sheva 84105, e-mail: elyager@bezeqint.net

PMC in a manageable time.

To overcome this problem, we propose a different sampling scheme, based on the *Stochastic Enumeration* (SE) method [18]. We show that the SE–based sampling is capable of handling rare–event probabilities while using a relatively small sample size. This will significantly boost our abilities to perform the reliability estimation in sense of precision and running time. While the SE belongs to the SIS family of algorithms, it is an extension of the well known Knuth's estimator [26] for approximating the cost of backtrack trees. The main difference between the corresponding Knuth's estimator and SE is that the latter has a budget parameter (B) that limits the number of parallel random walks. We show that this property has a crucial impact on the SE performance. In particular, it turn the SE into a *splitting* algorithm. It was shown that such splitting mechanisms can introduce a significant variance reduction [27], [28]. For a background on the splitting methods, see [15], [29], [30], [28].

The rest the paper is organized as follows. In Section II we give a short summary of the Spectra method and explain why it fails in the context of highly reliable networks. In Section III we give an introduction to the SE algorithm and show how the latter can be used to reliably estimate the Spectra. In Section IV we provide numerical evidence for the accuracy of our method. Finally, in Section V we summarize our findings and discuss possible directions for future research.

II. Spectra Method

We consider a special case of the terminal network reliability problem in which the nodes are completely reliable and all edge failure probabilities are equal; that is, $q_e = q$ for all $e \in E$. Under this simplified setting there exists an efficient approach to calculate the terminal network *reliability* $r(q)$ or *unreliability* $\overline{r}(q)$ — the so–called *Spectra* method. We now briefly describe the latter.

Let $e_1, e_2, \ldots, e_m$ be the network edges. Suppose that all of them are initially operational and thus the network is in the *UP* state, and let $\boldsymbol{\pi} = (e_{i_1}, \ldots, e_{i_m})$ be a permutation of edges. Given the permutation $\boldsymbol{\pi}$, start "erasing" edges (change the edges state from *up* to *down*) moving through the permutation from left to right and check the *UP/DOWN* state of the network after each step. Find the index j of the first edge e_{i_j}, $j = 1, \ldots, m$ for which the network switches from *UP* to *DOWN*. This index j is called the *anchor* of $\boldsymbol{\pi}$ and is denoted by $a(\boldsymbol{\pi})$.

Next, we assign the uniform distribution on the set of all edge permutations, that is, $\mathbb{P}(\boldsymbol{\Pi} = \boldsymbol{\pi}) = 1/m!$, and define the set

$$A(k) = \{\boldsymbol{\pi} \mid a(\boldsymbol{\pi}) = k\}.$$

Definition 2.1 (Destruction Spectra [22]): Let $\boldsymbol{\Pi}$ be a random permutation and define

$$f_k = \mathbb{P}(\boldsymbol{\Pi} \in A(k)) = \frac{|A(k)|}{m!}, \quad k = 0, \ldots, m.$$

Then,

$$S_p = \{f_0, f_1, \ldots, f_m\}$$

is called the *Destruction Spectra*, or simply *D–Spectra* of the network. □

Definition 2.2 (Cumulative D–Spectra [22]): The cumulative D–Spectra is defined by

$$C_{S_p} = \{F(0), F(1), \ldots, F(m)\},$$

where

$$F(k) = \sum_{i=0}^{k} f_i, \quad k = 0, \ldots, m.$$

□

The nice feature of the D–Spectra is that once C_{S_p} is available one can calculate directly the sought network unreliability $\overline{r}(q)$. Let us define a *failure set* to be an ordered set of edges such that their failure forces the network to enter the *DOWN* state and denote by $\mathcal{N}(k)$ the number of network failure sets of size k. Note that each such set is a collection of k edges whose failure results in the *DOWN* state of the network. It is readily seen that

$$\mathcal{N}(k) = \binom{m}{k} F(k). \tag{1}$$

This statement has a simple combinatorial explanation: $F(k)$ is the fraction of all failure sets of size k among all subsets of size k taken from the set of m components.

Moreover, note that the following holds.

- The network is *DOWN* if and only if it is in one of its failure sets.
- For fixed q each failure set of size k has the probability $q^k(1-q)^{m-k}$.

Combining this with (1) we obtain

$$\begin{aligned} \overline{r}(q) &= \sum_{k=0}^{m} \binom{m}{k} F(k) q^k (1-q)^{m-k} \\ &= \sum_{k=0}^{m} \mathcal{N}(k) q^k (1-q)^{m-k}. \end{aligned} \tag{2}$$

With this equation, it only remains to calculate the D–Spectra in order to calculate the network reliability for any q. As an example, consider the simple graph in Figure 1 and suppose that $K = \{s, t\}$.

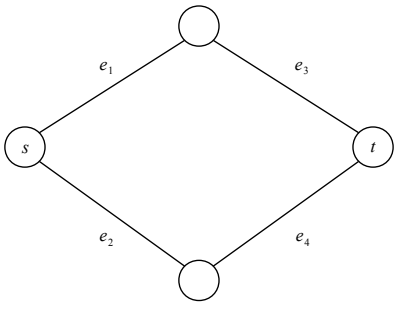

Fig. 1. A simple graph.

It is clear that zero or one edge removal cannot bring the network to the *DOWN* state so $f_0 = f_1 = 0$. In order

TABLE I
PERMUTATION–ANCHOR.

$\boldsymbol{\pi}$	$a(\boldsymbol{\pi})$	$\boldsymbol{\pi}$	$a(\boldsymbol{\pi})$	$\boldsymbol{\pi}$	$a(\boldsymbol{\pi})$
(e_1,e_2,e_3,e_4)	2	(e_2,e_3,e_4,e_1)	2	(e_3,e_4,e_2,e_1)	2
(e_1,e_2,e_4,e_3)	2	(e_2,e_3,e_1,e_4)	2	(e_3,e_4,e_1,e_2)	2
(e_1,e_3,e_2,e_4)	3	(e_2,e_4,e_3,e_1)	3	(e_4,e_1,e_2,e_3)	2
(e_1,e_3,e_4,e_2)	3	(e_2,e_4,e_1,e_3)	3	(e_4,e_1,e_3,e_2)	2
(e_1,e_4,e_2,e_3)	2	(e_3,e_1,e_2,e_4)	3	(e_4,e_2,e_3,e_1)	3
(e_1,e_4,e_3,e_2)	2	(e_3,e_1,e_4,e_2)	3	(e_4,e_2,e_1,e_3)	3
(e_2,e_1,e_4,e_3)	2	(e_3,e_2,e_4,e_1)	2	(e_4,e_3,e_2,e_1)	2
(e_2,e_1,e_3,e_4)	2	(e_3,e_2,e_1,e_4)	2	(e_4,e_3,e_1,e_2)	2

to calculate f_2, f_3 and f_4, consider Table I. From the data presented in Table I, the D–Spectra is given by

$$f_0 = f_1 = 0,\ f_2 = 16/24,\ f_3 = 8/24 \text{ and } f_4 = 0,$$

so we arrive at

$$S_p = \{0, 0, 2/3, 1/3, 0\} \text{ and } C_{S_p} = \{0, 0, 2/3, 1, 1\}.$$

As soon as we obtain this Spectra the network unreliability can be readily calculated for any q via (2).

Sometimes, it is more convenient to work with an equivalent Spectra object called the *Construction Spectra*. Under the construction settings, we start with all edges being in state *down*. In this case the network is clearly in the *DOWN* state too. Similarly to the D–Spectra let $\boldsymbol{\pi} = (e_{i_1}, \ldots, e_{i_m})$ be a permutation of the network edges. With this $\boldsymbol{\pi}$, start "building" edges (change the edges state from *down* to *up*) while moving through the permutation from left to right. The anchor of $\boldsymbol{\pi}$, $(a'(\boldsymbol{\pi}))$, is defined as the first index of the edge for which the network enters the *UP* state. Next, by specifying the set

$$A'(k) = \{\boldsymbol{\pi} \mid a'(\boldsymbol{\pi}) = k\},$$

the Construction Spectra (C–Spectra) S'_p and the Cumulative C–Spectra $C'_{S'_p}$ can be defined similarly to Definitions 2.1 and 2.2.

Definition 2.3 (Construction Spectra [22]): Let $\boldsymbol{\Pi}$ be a random permutation and define

$$f'_k = \mathbb{P}(\boldsymbol{\Pi} \in A'(k)) = \frac{|A'(k)|}{m!}, \quad k = 0, \ldots, m.$$

Then,

$$S'_p = \{f'_0, f'_1, \ldots, f'_m\}$$

is called the *Construction Spectra*, or simply *C–Spectra* of the network. □

Definition 2.4 (Cumulative Construction Spectra [22]): The cumulative C–Spectra is defined by

$$C'_{S'_p} = \{F'(0), F'(1), \ldots, F'(m)\},$$

where

$$F'(k) = \sum_{i=0}^{k} f'_i, \quad k = 0, \ldots, m.$$

□

Given the Construction Spectra the network *UP* probability is given by

$$\begin{aligned} r(q) &= \sum_{k=0}^{m} \binom{m}{k} F'(k) p^k (1-p)^{m-k} \\ &= \sum_{k=0}^{m} \mathcal{N}'(k) p^k (1-p)^{m-k}, \end{aligned}$$

where

$$\mathcal{N}'(k) = \binom{m}{k} F'(k).$$

Intuitively, both the destruction and the construction Spectra seems to share similar behavior; Proposition 2.1 establishes their equivalence.

Proposition 2.1 (Construction and Destruction Spectra equivalence): For the Construction and Destruction Spectra, it holds that

$$f_k = f'_{m-k} \quad \text{for all } k = 0, \ldots, m.$$

Proof: Let $\boldsymbol{\pi} = (e_{i_1}, \ldots, e_{i_m})$ be a permutation of edges and define the reverse permutation $\boldsymbol{\pi}^r$ as $\boldsymbol{\pi}^r = (e_{i_m}, \ldots, e_{i_1})$. Suppose that $\boldsymbol{\pi} \in A(k)$, that is $a(\boldsymbol{\pi}) = k$ for any $k = 0, \ldots, m$. Note that for this particular $\boldsymbol{\pi}$ the following holds.

- If the edges $e_{i_1}, \ldots, e_{i_k}$ and $e_{i_{k+1}}, \ldots, e_{i_m}$ are in *down* and *up* states respectively,the network is *DOWN*.
- If the edges $e_{i_1}, \ldots, e_{i_{k-1}}$ and $e_{i_k}, \ldots, e_{i_m}$ are in *down* and *up* states, the network is in *UP* state.

From this, $a'(\boldsymbol{\pi}^r) = m - k$, that is $\boldsymbol{\pi}^r \in A'(m-k)$. Hence, $\boldsymbol{\pi} \in A(k) \Rightarrow \boldsymbol{\pi}^r \in A'(m-k)$, and we conclude that $|A(k)| = |A'(m-k)|$, so, combining this with Definitions 2.1 and 2.3 the proof is completed by

$$f_k = \frac{|A(k)|}{m!} = \frac{|A'(m-k)|}{m!} = f'_{m-k}.$$

□

As an immediate consequence of Proposition 2.1, note that we can freely choose the Construction or the Destruction Spectra to work with and that this particular choice should be guided only by one's convenience. The above statement is summarized in Corollary 2.1.

Corollary 2.1: For any network with m edges, the D–Spectra can be transformed to C–Spectra and vice versa using the formula

$$f_k = f'_{m-k} \quad \text{for all } k = 0, \ldots, m.$$

□

Unfortunately, both the construction and the destruction Spectra are generally not available analytically and as consequence a Monte Carlo procedure should be applied. One of the widely adopted approaches, the PMC, is presented in Algorithm 2.1.

Algorithm 2.1 (PMC Algorithm): Given a graph $G(V, E, K)$ such that $|E| = m$ and a sample size N, execute the following steps.

1) **Step 1 (Initialization):** Set
$$\widehat{S_p} \equiv \{\widehat{f}_0, \ldots, \widehat{f_m}\} \longleftarrow \underbrace{\{0, 0, \ldots, 0\}}_{m+1}.$$
2) **Step 2 (Main loop):** Repeat N times.
   a) $\mathbf{\Pi} \leftarrow (e_{i_1}, \ldots, e_{i_m})$ (generate a random edge permutation).
   b) $k \leftarrow a(\mathbf{\Pi})$ (find the anchor).
   c) $\widehat{f}_k \longleftarrow \widehat{f}_k + 1$.
3) **Step 3 (Calculate D–Spectra):**
$$\widehat{f}_k \longleftarrow \frac{\widehat{f}_k}{N} \quad \text{for } k = 0, \ldots, m.$$
4) **Step 3 (Calculate Cumulative D–Spectra):**
$$\widehat{F(k)} \longleftarrow \sum_{i=0}^{k} \widehat{f}_i \quad \text{for } k = 0, \ldots, m.$$

□

Let us turn our attention to the Spectra estimation under rare event settings and consider the artificial Spectra in Table II. The left part of the table presents the "real" Spectra values (C_{S_p}) and the right stands for their estimates ($\widehat{C_{S_p}}$) using Algorithm 2.1. Note that the proposed Monte Carlo Algorithm 2.1 will not be able to estimate the left part of the Spectra $(F(4), F(5)$ and $F(6))$ in a reliable manner using any manageable sample size, say $N = 10^7$. Suppose that the PMC Algorithm 2.1 delivers the Spectra estimation presented in Table II.

TABLE II
ARTIFICIAL SPECTRA.

C_{S_p}				$\widehat{C_{S_p}}$			
$F(0)$	0	$F(10)$	0.3	$\widehat{F(0)}$	0	$\widehat{F(10)}$	0.3
$F(1)$	0	$F(11)$	0.5	$\widehat{F(1)}$	0	$\widehat{F(11)}$	0.5
$F(2)$	0	$F(12)$	0.7	$\widehat{F(2)}$	0	$\widehat{F(12)}$	0.7
$F(3)$	0	$F(13)$	0.9	$\widehat{F(3)}$	0	$\widehat{F(13)}$	0.9
$F(4)$	10^{-12}	$F(14)$	1	$\widehat{F(4)}$	0	$\widehat{F(14)}$	1
$F(5)$	10^{-10}	$F(15)$	1	$\widehat{F(5)}$	0	$\widehat{F(15)}$	1
$F(6)$	10^{-8}	$F(16)$	1	$\widehat{F(6)}$	0	$\widehat{F(16)}$	1
$F(7)$	10^{-6}	$F(17)$	1	$\widehat{F(7)}$	10^{-6}	$\widehat{F(17)}$	1
$F(8)$	10^{-3}	$F(18)$	1	$\widehat{F(8)}$	10^{-3}	$\widehat{F(18)}$	1
$F(9)$	0.1	$F(19)$	1	$\widehat{F(9)}$	0.1	$\widehat{F(19)}$	1

Applying (2) to the Spectra values from Table II, we get the corresponding network reliability estimations. Table III summarizes the values for the obtained estimators $\overline{r}(q)$ and $\widehat{\overline{r}(q)}$ using the exact and the approximated Spectra respectively.

One can easily note that the right column of Table III is a clear underestimation for $q \leqslant 0.1^4$. A more careful observation of the results reveals the importance of the leftmost values of

TABLE III
NETWORK *DOWN* PROBABILITIES FOR DIFFERENT VALUES OF q.

q	$\overline{r}(q)$	$\widehat{\overline{r}(q)}$
0.1^{10}	$3.88 \cdot 10^{-49}$	$5.04 \cdot 10^{-72}$
0.1^{9}	$3.88 \cdot 10^{-45}$	$5.04 \cdot 10^{-65}$
0.1^{8}	$3.88 \cdot 10^{-41}$	$5.04 \cdot 10^{-58}$
0.1^{7}	$3.88 \cdot 10^{-37}$	$5.04 \cdot 10^{-51}$
0.1^{6}	$3.88 \cdot 10^{-33}$	$5.05 \cdot 10^{-44}$
0.1^{5}	$3.89 \cdot 10^{-29}$	$5.11 \cdot 10^{-37}$
0.1^{4}	$3.99 \cdot 10^{-25}$	$5.00 \cdot 10^{-30}$
0.1^{3}	$5.37 \cdot 10^{-21}$	$1.34 \cdot 10^{-22}$
0.1^{2}	$1.62 \cdot 10^{-14}$	$1.58 \cdot 10^{-14}$
0.1^{1}	$4.71 \cdot 10^{-6}$	$4.71 \cdot 10^{-6}$

Spectra. Note that $\widehat{F(4)}, \widehat{F(5)}$ and $\widehat{F(6)}$ were "evaluated" to zero, and, for $q \leqslant 0.1^4$, (2) accumulates most of its mass for $k = 4, 5, 6$. For example, if we examine $q = 0.1^4$ and evaluate (2) for $k = 0, \ldots, 6$, we get

$$\sum_{k=0}^{6} \binom{19}{k} F(k)(0.1^4)^k(1 - 0.1^4)^{19-k} \approx 3.989017 \cdot 10^{-25}$$

which is almost equal to the true probability

$$\begin{aligned}\overline{r}\left(0.1^4\right) &= \sum_{k=0}^{19} \binom{19}{k} F(k)(0.1^4)^k(1 - 0.1^4)^{19-k} \\ &= 3.989067 \cdot 10^{-25}.\end{aligned}$$

Clearly, the remaining probability mass

$$\sum_{k=7}^{19} \binom{19}{k} \widehat{F(k)}(0.1^4)^k(1 - 0.1^4)^{19-k} \approx 5 \cdot 10^{-30}$$

forms the value of $\widehat{\overline{r}\,(0.1^4)}$ because $\widehat{F(k)} = F(k)$ for $k = 7, \ldots, 19$.

We shall summarize our findings as follows. For a highly reliable networks (that is, networks with small edge failure probability q), Algorithm 2.1 fails to produce a reliable result because the latter require calculations that involves an accurate estimation of rare event probabilities. To overcome this problem we propose to adopt the SE algorithm that is described in the next section.

III. STOCHASTIC ENUMERATION METHOD FOR COUNTING TREES

In this section we provide a brief overview of the SE method for counting trees. The algorithm is an extension of the one introduced in [28]. Our setting is as follows. Consider a rooted tree $T = (\mathcal{V}, \mathcal{E})$ with node set $\mathcal{V}$ and edge set $\mathcal{E}$ (so that $|\mathcal{E}| = |\mathcal{V}| - 1$). We denote the root of the tree by v_0, and for any $v \in \mathcal{V}$ the subtree rooted at v is denoted by T_v. Which each node v is associated a nonnegative cost $c(v)$. The main quantity of interest is the total cost of the tree,

$$\text{Cost}(T) = \sum_{v \in \mathcal{V}} c(v)$$

or, more generally, the total cost of a subtree T_v — denoted by $\mathrm{Cost}(T_v)$. For each node v we denote the set of successors of v by $S(v)$.

Definition 3.1 (Hyper nodes and forests): Let $\{v_1, \ldots, v_r\} \in \mathcal{V}$ be tree nodes.

- We call a collection of distinct nodes in the same level of the tree
$$\mathbf{v} = \{v_1, \ldots, v_r\}$$
a *hyper node* of cardinality $|\mathbf{v}| = r$.
- Let $\mathbf{v}$ be a hyper node. Generalizing the tree node cost, we define the cost of the hyper node as
$$c(\mathbf{v}) = \sum_{v \in \mathbf{v}} c(v).$$
- Let $\mathbf{v}$ be a hyper node. Define the set of successors of $\mathbf{v}$ as
$$S(\mathbf{v}) = \bigcup_{v \in \mathbf{v}} S(v).$$
- Let $\mathbf{v}$ be a hyper node and let $B \in \mathbb{N}$, $B \geqslant 1$. Define:
$$H(\mathbf{v}) = \begin{cases} \{S(\mathbf{v})\} & \text{if} \quad |S(\mathbf{v})| \leqslant B \\ \{\mathbf{w} \mid \mathbf{w} \subseteq S(\mathbf{v}),\ |\mathbf{w}| = B\} & \text{if} \quad |S(\mathbf{v})| > B. \end{cases}$$
- For each hyper node $\mathbf{v}$ let
$$T_{\mathbf{v}} = \bigcup_{v \in \mathbf{v}} T_v$$
be the forest of trees rooted at $\mathbf{v}$. See Figure 2 for an example of hyper node $\mathbf{v} = \{v_1, v_2, v_3, v_4\}$ and its corresponding forest $T_{\mathbf{v}} = \{T_{v_1}, T_{v_2}, T_{v_3}, T_{v_4}\}$.
- For each forest rooted at hypernode $\mathbf{v}$, define its total cost as
$$\mathrm{Cost}\,(T_{\mathbf{v}}) = \sum_{v \in \mathbf{v}} \mathrm{Cost}(T_v).$$

□

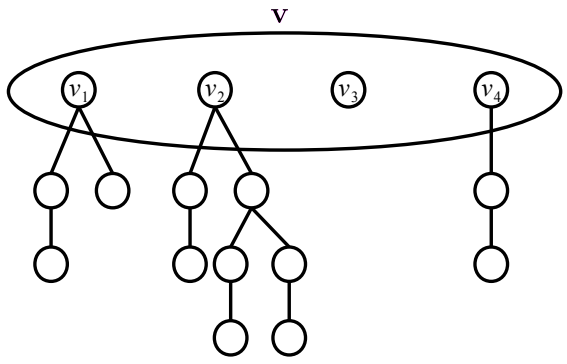

Fig. 2. Hyper node $\mathbf{v}$ that contains regular tree nodes v_1, v_2, v_3 and v_4 with their corresponding subtrees.

With these definitions in hand we are ready to state the main SE algorithm.

Algorithm 3.1 (Stochastic Enumeration Algorithm): Given a forest $T_{\mathbf{v}}$ of height h rooted at a hypernode $\mathbf{v}$, and a budget $B \geqslant 1$, execute the following steps.

1) **Step 1 (Initialization):** Set $k \leftarrow 0$, $D \leftarrow 1$, $\mathbf{X}_0 = \mathbf{v}$ and $C_{\mathrm{SE}} \leftarrow c(\mathbf{X}_0)/|\mathbf{X}_0|$.
2) **Step 2 (Compute the successors):** Let $S(\mathbf{X}_k)$ be the set of all children of $\mathbf{X}_k$.
3) **Step 3 (Terminal position?):** If $|S(\mathbf{X}_k)| = 0$, the algorithm stops, returning $|\mathbf{v}|\, C_{\mathrm{SE}}$ as an estimator of $\mathrm{Cost}(T_{\mathbf{v}})$.
4) **Step 4 (Advance):** Choose hyper node $\mathbf{X}_{k+1} \in H(\mathbf{X}_k)$ at random, each choice being equally likely. (Thus, each choice occurs with probability $1/|H(\mathbf{X}_k)|$.) Set $D_k = \frac{|S(\mathbf{X}_k)|}{|\mathbf{X}_k|}$ and $D \leftarrow D_k D$, then set $C_{\mathrm{SE}} \leftarrow C_{\mathrm{SE}} + \left(\frac{c(\mathbf{X}_{k+1})}{|\mathbf{X}_{k+1}|}\right) D$. Increase k by 1 and return to Step 2.

□

It can be shown that the output of Algorithm 3.1 is a random variable

$$C_{\mathrm{SE}} = \frac{c(\mathbf{X}_0)}{|\mathbf{X}_0|} + \frac{|S(\mathbf{X}_0)|}{|\mathbf{X}_0|}\frac{c(\mathbf{X}_1)}{|\mathbf{X}_1|} + \frac{|S(\mathbf{X}_0)|}{|\mathbf{X}_0|}\frac{|S(\mathbf{X}_1)|}{|\mathbf{X}_1|}\frac{c(\mathbf{X}_2)}{|\mathbf{X}_2|} + \cdots + \left(\prod_{0 \leqslant j \leqslant \tau-1} \frac{|S(\mathbf{X}_j)|}{|\mathbf{X}_j|}\right)\frac{c(\mathbf{X}_\tau)}{|\mathbf{X}_\tau|}, \tag{3}$$

where $\tau \leqslant h$ and

$$\left(\prod_{0 \leqslant j \leqslant k-1} \frac{|S(\mathbf{X}_j)|}{|\mathbf{X}_j|}\right)\frac{c(\mathbf{X}_k)}{|\mathbf{X}_k|}$$

is the cost of vertices at tree level k. Moreover, this estimator is unbiased, that is, for a tree T rooted at v_0, and for $\mathbf{v}_0 = \{v_0\}$,

$$\mathbb{E}\left(C_{\mathrm{SE}}(T_{\mathbf{v}_0})\right) = \mathrm{Cost}\,(T).$$

For the proof of unbiasedness and for typical run of Algorithm 3.1, see Appendices A and B respectively.

Note that (3) represents the sum of unbiased estimators to the costs of tree levels at heights $k = 0, \ldots, h$. With this observation in hand one can introduce a simple modification of Algorithm 3.1 such that instead of the entire tree cost, the algorithm will return separate estimators for each level. All we need to do is to replace the single random variable C_{SE} with h variables $C_{\mathrm{SE}}^{(0)}, \ldots, C_{\mathrm{SE}}^{(h)}$, and substitute the expression $C_{\mathrm{SE}} \leftarrow C_{\mathrm{SE}} + \left(\frac{c(\mathbf{X}_{k+1})}{|\mathbf{X}_{k+1}|}\right) D$ with $C_{\mathrm{SE}}^{(k+1)} \leftarrow C_{\mathrm{SE}}^{(k+1)} + \left(\frac{c(\mathbf{X}_{k+1})}{|\mathbf{X}_{k+1}|}\right) D$ in the last step of Algorithm 3.1.

Next, we concentrate on the crucial property of the SE algorithm — the built–in *splitting* mechanism.

A. SE Splitting mechanism

The SE algorithm posses a built–in *splitting* mechanism. The latter can bring enormous variance reduction, [11], [14], [29], [28].

Consider the "hair brush" tree T in Figure 3 and suppose that the cost of all vertices is zero except for v_{n+1}, which has a cost of unity. Our goal is to estimate the cost of this tree, which obviously satisfies

$$\mathrm{Cost}(T) = 1.$$

It will become clear from the following discussion that the budget parameter B is controlling the SE Algorithm 3.1 splitting capability. We will consider two cases. In particular we examine the behavior of the SE Algorithm 3.1 with budgets $B = 1$ and $B = 2$ respectively.

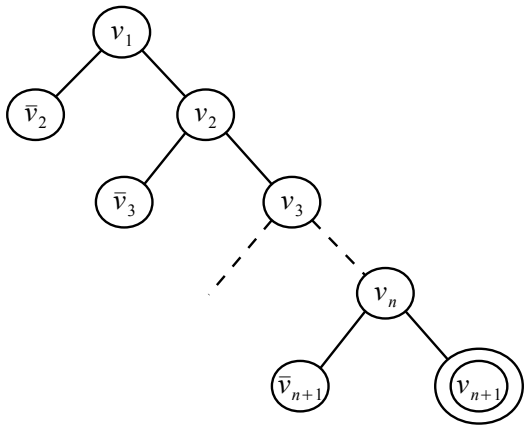

Fig. 3. The hair brush tree.

- If we set $B = 1$, the SE Algorithm 3.1 essentially adopts the behaviour of Knuth's estimator, [26]. Note that in this case the algorithm reaches the vertex of interest, v_{n+1}, with probability $1/2^n$ and with $D = 2^n$. In all other cases, the algorithm terminates with some D' and a zero cost node $\bar{v}_i$, $i = 2, \ldots, n+1$. It follows that the expectation and variance of the corresponding SE estimator are

$$\mathbb{E}\left(C_{\mathrm{SE}}\right) = \frac{1}{2^n} \cdot 2^n \cdot 1 + \frac{2^n - 1}{2^n} \cdot D' \cdot 0 = 1,$$

and

$$\mathbb{E}\left(C_{\mathrm{SE}}^2\right) = \frac{1}{2^n} \cdot (2^n \cdot 1)^2 + \frac{2^n - 1}{2^n} \cdot (D' \cdot 0)^2 = 2^n \;\Rightarrow$$
$$\Rightarrow\; \mathrm{Var}\left(C_{\mathrm{SE}}\right) = \mathbb{E}\left(C_{\mathrm{SE}}^2\right) - \mathbb{E}\left(C_{\mathrm{SE}}\right)^2 = 2^n - 1.$$

- On the other hand, setting $B = 2$ will force Algorithm 3.1 to reach v_{n+1} with probability 1. Note that with this budget and for $i = 2, \ldots, n$, one algorithm trajectory, (random walk from the tree root), is always disappearing at $\bar{v}_i$ vertices from the left, but the second one is always split in two new trajectories at the corresponding v_i nodes from the right. Following the execution steps of the SE Algorithm 3.1 one can verify that at the final iteration the cost of the hyper node $\mathbf{X}_{n+1} = \{\bar{v}_{n+1},\ v_{n+1}\}$ is $0 + 1 = 1$, so

$$\left(\frac{c(\mathbf{X}_{n+1})}{|\mathbf{X}_{n+1}|}\right) = \frac{1}{2}.$$

In addition, the final value of D is

$$D = 2 \cdot \underbrace{1 \cdots 1}_{n-1 \text{ times}} = 2.$$

It follows that the expectation and variance of the corresponding SE estimator are

$$\mathbb{E}\left(C_{\mathrm{SE}}\right) = 1 \cdot 2 \cdot \frac{1}{2} = 1,$$

and

$$\mathbb{E}\left(C_{\mathrm{SE}}^2\right) = 1 \cdot \left(2 \cdot \frac{1}{2}\right)^2 = 1 \;\Rightarrow$$
$$\Rightarrow\; \mathrm{Var}\left(C_{\mathrm{SE}}\right) = \mathbb{E}\left(C_{\mathrm{SE}}^2\right) - \mathbb{E}\left(C_{\mathrm{SE}}\right)^2 1 - 1 = 0.$$

By increasing the budget B from 1 to 2 we managed to achieve remarkable variance reduction, from $2^n - 1$ to zero. Clearly, we presented an artificial example but it is also illustrative enough for our purposes. Generally speaking, by increasing the budget we cannot expect to obtain a zero variance estimator for hard approximation problems, but we do hope to achieve a significant variance reduction.

B. SE for Network Reliability

To start with, consider the set of all edge permutations. With these permutations it is possible to build a tree object that we call the *permutation tree*, for which each path from the root to a leaf corresponds to some specific permutation $\boldsymbol{\pi}$. For example, consider the simple graph in Figure 1 and the corresponding permutation tree that is presented in Figure 4.

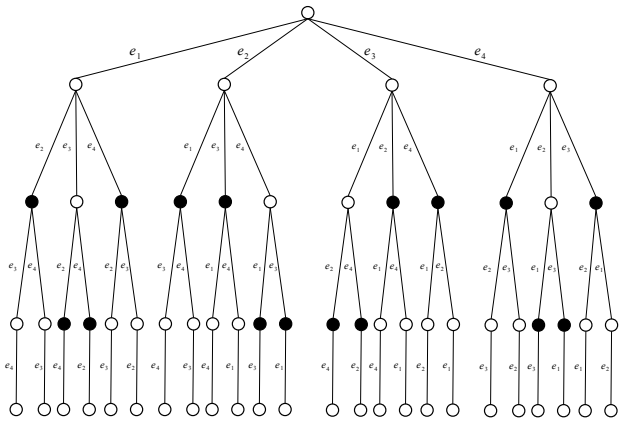

Fig. 4. The simple graph permutation tree.

By definition, each path from the root to a leaf corresponds to unique edge permutation $\boldsymbol{\pi}$ and each black vertex corresponds to this permutation's anchor, $a(\boldsymbol{\pi})$. It is not hard to see by following the steps of Algorithm 2.1 that step $2(a)$ generates a random tree path ($\boldsymbol{\pi}$) and step $2(b)$ finds the anchor $a(\boldsymbol{\pi})$ — the black vertex. Recall that by definition,

$$f_k = \frac{|A(k)|}{m!},$$

so, we can use Figure 4 to calculate the Spectra. The latter is accomplished as follows. Note that we have 8 anchor vertices at level 2 and each such vertex induces two additional paths to the leaves, so,

$$f_2 = \frac{8 \cdot 2}{4!} = \frac{2}{3}.$$

Having in mind that at the third tree level the vertices induce a single path to the leaf and combining this with the fact that we have 8 anchor vertices at this level, we conclude that

$$f_3 = \frac{8 \cdot 1}{4!} = \frac{1}{3}.$$

The other Spectra components are equal to zero, so by Definition 2.2,

$$S_p = \left\{0, 0, \frac{2}{3}, \frac{1}{3}, 0\right\},$$

that is,

$$F(0) = 0,\ F(1) = 0,\ F(2) = \frac{2}{3},\ F(3) = 1,\ F(4) = 1.$$

With the above Cumulative Spectra we get a full network characterization and the desired unreliability $\overline{r}(q)$ can be calculated for any q using (2).

The above discussion is generalized in Theorem 3.1, which provides the connection between the number of anchor (black) nodes at each tree level and the corresponding Spectra.

Theorem 3.1 (Spectra as a function of anchor nodes): Let T be a permutation tree. Then,

$$f_k = \frac{\text{\# of anchor nodes at tree level } k}{\text{total \# of nodes at level } k}.$$

Proof: Suppose that there are m edges and let $L(k)$ be the set of anchor nodes at tree level k and consider the path from the tree root to some $v \in L(k)$. Clearly, an anchor node cannot appear on this path at level j such that $j < k$ or $j > k$, because in this case $v \notin L(k)$, so,

$$|A(k)| = |L(k)| \cdot (\text{\# of induced paths from } v \in L(k)).$$

Having in mind that each node at level k induces

$$(m-k)(m-(k+1)) \cdots (m-(m-1)) = (m-k)!$$

paths to the leaves, we arrive at

$$|A(k)| = |L(k)|(m-k)!.$$

Moreover, the total number of (anchor and non–anchor) nodes at tree level k is equal to

$$m(m-1)(m-2)\cdots(m-k+1) = \frac{m!}{(m-k)!},$$

so, from Definition 2.1,

$$f_k = \frac{|A(k)|}{m!} = \frac{|L(k)|(m-k)!}{m!} = \frac{|L(k)|}{\frac{m!}{(m-k)!}} = \\ = \frac{\text{\# of anchor nodes at tree level } k}{\text{total \# of nodes at level } k}.$$

□

All we need to do now is to assign the cost of unity to the anchor nodes and the zero cost to non–anchor vertices. Recall the Algorithm 3.1 and note that $C_{\text{SE}}^{(k)}$ for $k = 0, \ldots, m$ is an unbiased estimator to the cost of level k. Then, from Theorem 3.1 the D–Spectra is given by

$$f_k = \mathbb{E}\left(C_{\text{SE}}^{(k)}\right) \Big/ \frac{m!}{(m-k)!}.$$

Theorem 3.1 opens the way for Spectra calculation using the SE sampling scheme. The algorithm to do so may be summarized as follows.

Algorithm 3.2 (SE sampling scheme): Given a graph $G(V, E, K)$ such that $|E| = m$, the budget B and a sample size N, execute the following steps.

1) **Step 1 (SE):** Run the SE Algorithm 3.1 to estimate
$$\widehat{C_{\text{SE}}^{(0)}}, \ldots, \widehat{C_{\text{SE}}^{(h)}}.$$
2) **Step 2 (Estimate Spectra):** Set
$$\widehat{f_k} = \widehat{C_{\text{SE}}^{(k)}} \Big/ \frac{m!}{(m-k)!} \quad \text{for } k = 0, \ldots, m.$$
3) **Step 3 (Estimate Cumulative Spectra):**
$$\widehat{F(k)} \longleftarrow \sum_{i=0}^{k} \widehat{f_i} \quad \text{for } k = 0, \ldots, m.$$

□

It is important to note that the same permutation tree is used for both Construction and Destruction Spectra except that the anchor nodes will be placed in a different locations according to the adopted construction or the destruction setting respectively. Having this in mind, it follows that Theorem 3.1 is also valid for both D–Spectra and C–Spectra and, as a consequence, Algorithm 3.2 can be executed in both (construction/destruction) modes. If one chooses to calculate the C–Spectra, ***Step 2*** of Algorithm 3.2 should be written as

$$\widehat{f_{m-k}} = \widehat{C_{\text{SE}}^{(k)}} \Big/ \frac{m!}{(m-k)!} \quad \text{for } k = 0, \ldots, m.$$

Finally, recall the reliability estimation complication in the context of the highly reliable networks that we reported in Section II. We showed that the error is due to the PMC estimation problem of very small probabilities in the leftmost parts of the D–Spectra. First of all, note that those problematic components of D–Spectra correspond to the anchor nodes at the upper tree levels $(0, 1, 2, \ldots)$ of the permutation tree (in the destruction setting) or to the lower levels of the tree, $(m, m-1, m-2, \ldots)$ (in the construction setting); see Corollary 2.1.

The above discussion and the splitting example gives us a clue of how to use SE for our purposes. The SE's splitting mechanism is able to sample rare event anchor nodes in the final tree levels. Having this in mind, we will prefer to work with the C–Spectra and use Corollary 2.1 to obtain the corresponding D–Spectra. An additional technical point that one should be aware of is the structure of the permutation tree. In particular, note that as soon as an anchor node is discovered, its induced paths may readily be ignored. Indeed, there is no point in wasting resources by continuing to explore

such permutations. The corresponding truncated permutation tree is presented in Figure 5.

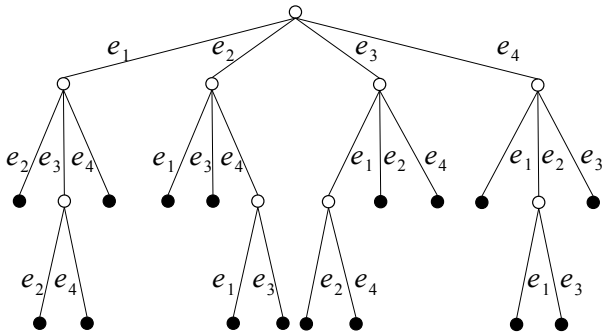

Fig. 5. The truncated permutation tree.

IV. Numerical results

In this section we present a numerical comparison between the PMC Algorithm 2.1 and the SE Algorithm 3.2 for different models. The results were obtained using an Intel Core i7 machine with 4 GB of RAM.

Model 1: Consider the graph family $\Psi(k,m),\ k\in\mathbb{N},\ m\geqslant 2k$ in Figure 6. Each graph is determined by two numbers k and m. In particular, this graph has two terminal nodes s and t, and k special vertices numbered $1,\ldots,k$ such that both s and t are connected to each of these vertices. In addition, both terminal vertices are connected to $m/2-k$ non–special vertices.

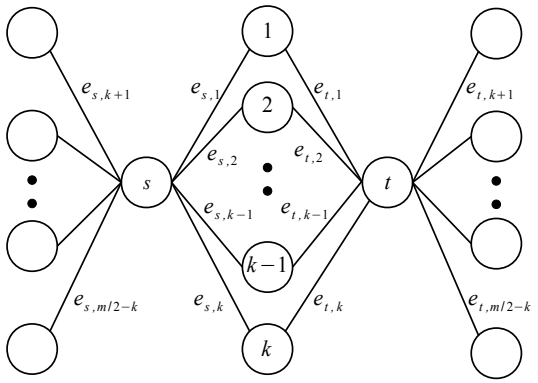

Fig. 6. The $\Psi(k,m)$ graph with $m-k+2$ vertices and m edges.

Despite the fact that $\Psi(k,m)$ is fabricated, it will be useful for our discussion. First of all it is straightforward to calculate the network unreliability. Clearly, the only edges that can cause the terminal disconnection are $e_{s,1},\ldots,e_{s,k},e_{t,1},\ldots,e_{t,k}$. So, the graph unreliability can be obtained by full enumeration of *up* and *down* states because one need only to check 2^k possibilities. An additional nice property is that the $f_0,\ldots,f_{k-1}$ are zero and

$$f_k=\mathbb{P}\left(a(\mathbf{\Pi})=k\right),$$

can be calculated analytically. To see this, consider a permutation $\boldsymbol{\pi}=(e_{i_1},\ldots,e_{i_m})$. Clearly, if $a(\boldsymbol{\pi})=k$, then $\{e_{i_1},\ldots,e_{i_k}\}$ should be a subset of $\{e_{s,1},\ldots,e_{s,k},e_{t,1},\ldots,e_{t,k}\}$. The latter occurs with probability

$$\frac{\binom{2k}{k}k!(m-k)!}{m!}.$$

And now, given that $\{e_{i_1},\ldots,e_{i_k}\}\subset\{e_{s,1},\ldots,e_{s,k},e_{t,1},\ldots,e_{t,k}\}$, we shall decide about the way to pick k edges such that the terminals will become disconnected. For the first edge there are $2k$ possibilities and suppose without loss of generality that we picked $e_{s,1}$. Now we need to pick the second edge; note that it cannot be $e_{t,1}$, because if we choose to disconnect it, there is no way to pick additional $k-2$ edges to disconnect the terminals. This means that the choice is from $2k-2$ possibilities and so on, so, we end up with

$$\frac{2k(2k-2)\cdots 2}{\binom{2k}{k}k!}$$

and arrive at

$$f_k=\mathbb{P}\left(a(\boldsymbol{\pi})=k\right)=\frac{\binom{2k}{k}k!(m-k)!}{m!}\cdot\frac{2k(2k-2)\cdots 2}{\binom{2k}{k}k!}= \tag{4}$$
$$=\frac{(2k(2k-2)\cdots 2)\,(m-k)!}{m!}.$$

Tables IV and V summarize 10 typical runs of PMC Algorithm 2.1 versus SE Algorithm 3.2 for several $\Psi(k,m)$ models. We can observe from those tables that PMC is unable to estimate the Spectra value f_k. On the other hand, the SE is not sensitive to this rare event probability. For the SE Algorithm 3.2, we took a budget $B=10$ and a sample size of $N=1000$. For the PMC Algorithm 2.1, we adjusted the sample size such that the CPU is comparable to the one of SE.

Note that for each $\Psi(k,m)$ model we get the same probability of network failure because they share the same parameter k. In particular, we set $q=10^{-4}$ and calculate the exact unreliability via full enumeration. The analytical probability of the failure is $\overline{r}\left(10^{-4}\right)=1.02\cdot 10^{-37}$. The analytical f_k was obtained using (4).

Model 2: It is worth nothing that PMC is able to deliver accurate estimations for small failure probabilities. The purpose of this example is to demonstrate that one can construct a graph instance such that the Spectra does not have very small components and show that the PMC can estimate it with arbitrary q values (in particular for $q\to 0^+$).

For this experiment we take the dodecahedron graph, the model that is widely considered for a network reliability examples. We show that the problem is easy in sense that there are no rare events involved; that is, we expect both the

TABLE IV
PMC SPECTRA ESTIMATION FOR $\Psi(k,m)$.

		PMC			
$\Psi(k,m)$	f_k	$\widehat{f_k}$	$\widehat{\overline{r}(10^{-4})}$	CPU	N
$\Psi(10,20)$	$5.54\cdot10^{-3}$	$5.62\cdot10^{-3}$	$1.04\cdot10^{-37}$	2.48	$2.0\cdot10^{5}$
$\Psi(10,40)$	$1.21\cdot10^{-6}$	$2.01\cdot10^{-6}$	$1.69\cdot10^{-37}$	6.49	$2.0\cdot10^{5}$
$\Psi(10,60)$	$1.36\cdot10^{-8}$	0	$1.05\cdot10^{-42}$	23.1	$4.0\cdot10^{5}$
$\Psi(10,80)$	$6.22\cdot10^{-10}$	0	$1.00\cdot10^{-47}$	31.7	$4.5\cdot10^{5}$
$\Psi(10,100)$	$5.92\cdot10^{-11}$	0	$2.67\cdot10^{-53}$	53.3	$5.0\cdot10^{5}$

TABLE V
SE SPECTRA ESTIMATION FOR $\Psi(k,m)$.

		SE		
$\Psi(k,m)$	f_k	$\widehat{f_k}$	$\widehat{\overline{r}(10^{-4})}$	CPU
$\Psi(10,20)$	$5.54\cdot10^{-3}$	$5.56\cdot10^{-3}$	$1.03\cdot10^{-37}$	2.18
$\Psi(10,40)$	$1.21\cdot10^{-6}$	$1.24\cdot10^{-6}$	$1.05\cdot10^{-37}$	8.69
$\Psi(10,60)$	$1.36\cdot10^{-8}$	$1.47\cdot10^{-8}$	$1.11\cdot10^{-37}$	18.4
$\Psi(10,80)$	$6.22\cdot10^{-10}$	$5.27\cdot10^{-10}$	$8.68\cdot10^{-38}$	31.9
$\Psi(10,100)$	$5.92\cdot10^{-11}$	$6.06\cdot10^{-11}$	$1.05\cdot10^{-37}$	51.1

PMC and the SE algorithms to report reliable results. We set the terminal set to be $K=\{1,20\}$.

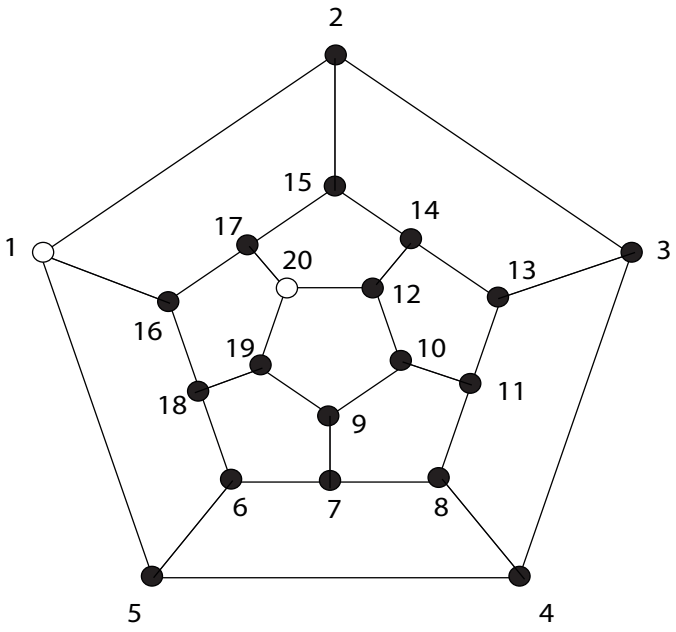

Fig. 7. The dodecahedron graph.

We run the SE algorithm with $N=1000$ and $B=10$ and the Spectra Algorithm with $N=100000$. The running time for SE and Spectra is 4.76 and 1.16 seconds respectively. The problem is easy in sense that there are no rare event probabilities involved in the Spectra estimation, that is, the minimal component has the value $\approx 5\cdot10^{-4}$; so, both algorithms deliver excellent performances in a reasonable time. Tables VI and VII provides the estimated Spectra and the corresponding probability of the network to be in *DOWN* state calculations.

TABLE VI
PMC SPECTRA ESTIMATION FOR THE DODECAHEDRON GRAPH.

	SE	PMC		SE	PMC
$F(1)$	0	0	$F(16)$	$7.90\cdot10^{-1}$	$7.91\cdot10^{-1}$
$F(2)$	0	0	$F(17)$	$8.48\cdot10^{-1}$	$8.49\cdot10^{-1}$
$F(3)$	$5.01\cdot10^{-4}$	$4.90\cdot10^{-4}$	$F(18)$	$8.91\cdot10^{-1}$	$8.93\cdot10^{-1}$
$F(4)$	$2.22\cdot10^{-3}$	$2.18\cdot10^{-3}$	$F(19)$	$9.24\cdot10^{-1}$	$9.26\cdot10^{-1}$
$F(5)$	$6.30\cdot10^{-3}$	$6.33\cdot10^{-3}$	$F(20)$	$9.49\cdot10^{-1}$	$9.51\cdot10^{-1}$
$F(6)$	$1.41\cdot10^{-2}$	$1.43\cdot10^{-2}$	$F(21)$	$9.67\cdot10^{-1}$	$9.68\cdot10^{-1}$
$F(7)$	$2.90\cdot10^{-2}$	$2.86\cdot10^{-2}$	$F(22)$	$9.80\cdot10^{-1}$	$9.80\cdot10^{-1}$
$F(8)$	$5.53\cdot10^{-2}$	$5.45\cdot10^{-2}$	$F(23)$	$9.89\cdot10^{-1}$	$9.88\cdot10^{-1}$
$F(9)$	$9.80\cdot10^{-2}$	$9.68\cdot10^{-2}$	$F(24)$	$9.94\cdot10^{-1}$	$9.94\cdot10^{-1}$
$F(10)$	$1.65\cdot10^{-1}$	$1.65\cdot10^{-1}$	$F(25)$	$9.97\cdot10^{-1}$	$9.97\cdot10^{-1}$
$F(11)$	$2.62\cdot10^{-1}$	$2.60\cdot10^{-1}$	$F(26)$	$9.99\cdot10^{-1}$	$9.99\cdot10^{-1}$
$F(12)$	$3.81\cdot10^{-1}$	$3.79\cdot10^{-1}$	$F(27)$	1	1
$F(13)$	$5.03\cdot10^{-1}$	$5.03\cdot10^{-1}$	$F(28)$	1	1
$F(14)$	$6.19\cdot10^{-1}$	$6.18\cdot10^{-1}$	$F(29)$	1	1
$F(15)$	$7.15\cdot10^{-1}$	$7.15\cdot10^{-1}$	$F(30)$	1	1

TABLE VII
SE SPECTRA ESTIMATION FOR THE DODECAHEDRON GRAPH.

	SE	PMC
q	$\widehat{\overline{r}(q)}$	$\widehat{\overline{r}(q)}$
$1.00\cdot10^{-5}$	$2.03\cdot10^{-15}$	$1.99\cdot10^{-15}$
$1.00\cdot10^{-4}$	$2.04\cdot10^{-12}$	$1.99\cdot10^{-12}$
$1.00\cdot10^{-3}$	$2.04\cdot10^{-9}$	$2.00\cdot10^{-9}$
$1.00\cdot10^{-2}$	$2.10\cdot10^{-6}$	$2.05\cdot10^{-6}$
$1.00\cdot10^{-1}$	$2.91\cdot10^{-3}$	$2.89\cdot10^{-3}$
$2.00\cdot10^{-1}$	$3.53\cdot10^{-2}$	$3.51\cdot10^{-2}$
$3.00\cdot10^{-1}$	$1.59\cdot10^{-1}$	$1.58\cdot10^{-1}$
$4.00\cdot10^{-1}$	$3.97\cdot10^{-1}$	$3.96\cdot10^{-1}$
$5.00\cdot10^{-1}$	$6.61\cdot10^{-1}$	$6.61\cdot10^{-1}$
$6.00\cdot10^{-1}$	$8.51\cdot10^{-1}$	$8.52\cdot10^{-1}$
$7.00\cdot10^{-1}$	$9.49\cdot10^{-1}$	$9.49\cdot10^{-1}$
$8.00\cdot10^{-1}$	$9.88\cdot10^{-1}$	$9.88\cdot10^{-1}$

Model 3: For our last model we consider the hypercube graph H_5, and $K=\{1,25\}$. In graph theory, the hypercube graph H_n is a regular graph with 2^n vertices and $n2^{n-1}$ edges, [31]. In order to construct a hypercube graph, label every 2^n vertices with n-bit binary numbers and connect two vertices by an edge whenever the Hamming distance of their labels is 1.

We run the SE Algorithm 3.2 with $N=1000$ and $B=10$ and the PMC Algorithm 2.1 with $N=300000$. The running time for both SE and Spectra is 28 seconds. Table VIII provides the estimated network unreliability with both PMC and SE algorithms.

Note the difference in the *DOWN* probabilities for small values of q. We already saw this problem — the PMC Algorithm 2.1 cannot estimate rare event probabilities, hence, the delivered estimator is an under estimation. Table IX summarizes the first components of the obtained Spectra with PMC and SE algorithms respectively. (We do not report

TABLE VIII
PMC AND SE BASED NETWORK RELIABILITY ESTIMATIONS FOR H_5 GRAPH.

	SE	PMC
q	$\widehat{\overline{r}(q)}$	$\widehat{\overline{r}(q)}$
$1.00 \cdot 10^{-5}$	$2.15 \cdot 10^{-25}$	$2.32 \cdot 10^{-39}$
$1.00 \cdot 10^{-4}$	$2.15 \cdot 10^{-20}$	$2.31 \cdot 10^{-30}$
$1.00 \cdot 10^{-3}$	$2.13 \cdot 10^{-15}$	$2.20 \cdot 10^{-21}$
$1.00 \cdot 10^{-2}$	$2.05 \cdot 10^{-10}$	$1.38 \cdot 10^{-12}$
$1.00 \cdot 10^{-1}$	$1.91 \cdot 10^{-5}$	$2.07 \cdot 10^{-5}$
$2.00 \cdot 10^{-1}$	$6.20 \cdot 10^{-4}$	$7.09 \cdot 10^{-4}$
$3.00 \cdot 10^{-1}$	$5.07 \cdot 10^{-3}$	$5.50 \cdot 10^{-3}$
$4.00 \cdot 10^{-1}$	$2.49 \cdot 10^{-2}$	$2.55 \cdot 10^{-2}$
$5.00 \cdot 10^{-1}$	$9.45 \cdot 10^{-2}$	$9.43 \cdot 10^{-2}$
$6.00 \cdot 10^{-1}$	$2.95 \cdot 10^{-1}$	$2.94 \cdot 10^{-1}$
$7.00 \cdot 10^{-1}$	$6.35 \cdot 10^{-1}$	$6.36 \cdot 10^{-1}$
$8.00 \cdot 10^{-1}$	$8.85 \cdot 10^{-1}$	$8.86 \cdot 10^{-1}$

$F(0), \ldots, F(3)$, because they are estimated as zero by both methods).

TABLE IX
ESTIMATED SPECTRA COMPONENTS.

Algorithm	$F(4)$	$F(5)$	$F(6)$	$F(7)$	$F(8)$	$F(9)$
SE	$8.93 \cdot 10^{-8}$	$4.96 \cdot 10^{-7}$	$1.67 \cdot 10^{-6}$	$4.42 \cdot 10^{-6}$	$9.60 \cdot 10^{-6}$	$2.03 \cdot 10^{-5}$
PMC	0	0	0	0	$1.00 \cdot 10^{-5}$	$2.67 \cdot 10^{-5}$

Next, we run the PMC algorithm 2.1 with $10^6, 10^7$ and 10^8 sample sizes. We suspect that $N = 10^8$ this is enough to estimate the small probabilities. The results are summarized in Table X.

TABLE X
PMC SPECTRA ESTIMATION WITH DIFFERENT SAMPLE SIZES.

	$N = 10^6$	$N = 10^7$	$N = 10^8$
q	$\widehat{\overline{r}(q)}$	$\widehat{\overline{r}(q)}$	$\widehat{\overline{r}(q)}$
$1.00 \cdot 10^{-5}$	$1.39 \cdot 10^{-39}$	$9.01 \cdot 10^{-29}$	$2.40 \cdot 10^{-25}$
$1.00 \cdot 10^{-4}$	$1.38 \cdot 10^{-30}$	$8.98 \cdot 10^{-23}$	$2.40 \cdot 10^{-20}$
$1.00 \cdot 10^{-3}$	$1.32 \cdot 10^{-21}$	$8.71 \cdot 10^{-17}$	$2.39 \cdot 10^{-15}$
$1.00 \cdot 10^{-2}$	$8.40 \cdot 10^{-13}$	$6.70 \cdot 10^{-11}$	$2.27 \cdot 10^{-10}$
$1.00 \cdot 10^{-1}$	$1.74 \cdot 10^{-5}$	$1.98 \cdot 10^{-5}$	$2.03 \cdot 10^{-5}$
$2.00 \cdot 10^{-1}$	$6.64 \cdot 10^{-4}$	$6.58 \cdot 10^{-4}$	$6.63 \cdot 10^{-4}$
$3.00 \cdot 10^{-1}$	$5.39 \cdot 10^{-3}$	$5.37 \cdot 10^{-3}$	$5.38 \cdot 10^{-3}$
$4.00 \cdot 10^{-1}$	$2.56 \cdot 10^{-2}$	$2.56 \cdot 10^{-2}$	$2.56 \cdot 10^{-2}$
$5.00 \cdot 10^{-1}$	$9.50 \cdot 10^{-2}$	$9.48 \cdot 10^{-2}$	$9.48 \cdot 10^{-2}$
$6.00 \cdot 10^{-1}$	$2.95 \cdot 10^{-1}$	$2.94 \cdot 10^{-1}$	$2.94 \cdot 10^{-1}$
$7.00 \cdot 10^{-1}$	$6.36 \cdot 10^{-1}$	$6.36 \cdot 10^{-1}$	$6.36 \cdot 10^{-1}$
$8.00 \cdot 10^{-1}$	$8.86 \cdot 10^{-1}$	$8.86 \cdot 10^{-1}$	$8.86 \cdot 10^{-1}$

The CPU time was 77, 846 and 9000 seconds for $N = 10^6, N = 10^7$ and $N = 10^8$ sample size respectively. Note that only for $N = 10^8$, we manage to estimate the rare event probabilities. The SE does this in 28 seconds compared to 9000 seconds of PMC.

V. CONCLUSIONS

In this paper we established that when using the network signature method (Spectra), a satisfactory estimation of network *DOWN* state probability depends on the accurate evaluation of all Spectra components. In particular, the leftmost part of D–Spectra turns out to be critical when dealing with highly reliable networks. We showed that under this setting, the simple PMC algorithm for estimating the network unreliability fails to provide a reliable answer. To successfully overcome this issue we introduced an adaptation of new sampling scheme for tree cost evaluation — Stochastic Enumeration (SE). Our numerical results indicate that thanks to its built–in *splitting* mechanism SE is able to successfully work in the rare event environment while using a relatively small sample size. Having in mind that some rare event estimation problems can be represented as trees, we conjecture that the SE may be viewed as a general powerful technique for rare event probability estimation and we expect to see its further usage in the future.

APPENDIX A
PROOF OF SE'S UNBIASEDNESS

To prove the main result we will need the following technical lemma.

Lemma A.1 (Sum of k-subsets): Let $r_1, \ldots, r_n$ and u be scalars such that

$$\forall j = 1, \ldots, n, \;\; r_j \geqslant 0 \text{ and } u = 1, \ldots, n,$$

and,

$$S = \{s \mid s \subseteq \{r_1, \ldots, r_n\}, |s| = u\}.$$

Then,

$$\sum_{s \in S} \frac{\sum_{r \in s} r}{u} = \frac{\binom{n}{u} \sum_{1 \leqslant j \leqslant n} r_j}{n}$$

holds.

Proof: Note that there are exactly $\binom{n-1}{u-1}$ subsets in which each r_j $(j = 1, \ldots, n)$ appears. We conclude the proof with

$$\sum_{s \in S} \frac{\sum_{r \in s} r}{u} = \frac{\binom{n-1}{u-1}}{u} \sum_{1 \leqslant j \leqslant n} r_j = \frac{\frac{(n-1)!}{(u-1)!(n-u)!}}{u} \sum_{1 \leqslant j \leqslant n} r_j$$
$$= \frac{\frac{n!}{u!(n-u)!}}{n} \sum_{1 \leqslant j \leqslant n} r_j = \frac{\binom{n}{u} \sum_{1 \leqslant j \leqslant n} r_j}{n}.$$

□

Recall from Definition 3.1 that for any hypernode $\mathbf{v}$, $T_{\mathbf{v}}$ denotes the forest rooted at $\mathbf{v}$, and that its cost is $\mathrm{Cost}(T_{\mathbf{v}})$. Let $|\mathbf{v}|\, C_{\mathrm{SE}}(T_{\mathbf{v}})$ be the corresponding estimator, as returned by Algorithm 3.1. The following theorem shows that this estimator is unbiased.

Theorem A.1 (Unbiased Estimator): Let $\mathbf{v}$ be a hyper node and let $H(S(\mathbf{v})) = \{\mathbf{w}_1, \ldots, \mathbf{w}_d\}$ be its set of hyper children. Then,

$$\mathbb{E}(C_{\mathrm{SE}}(T_{\mathbf{v}})) = \frac{\mathrm{Cost}(T_{\mathbf{v}})}{|\mathbf{v}|}. \tag{5}$$

Proof: By the recursive structure of Algorithm 3.1, we have

$$C_{\mathrm{SE}}\left(T_{\mathbf{v}}\right)=\frac{c(\mathbf{v})}{|\mathbf{v}|}+\frac{|S(\mathbf{v})|}{|\mathbf{v}|}C_{\mathrm{SE}}\left(T_{\mathbf{W}}\right), \tag{6}$$

where $\mathbf{W}$ is a hyperchild of $\mathbf{v}$ selected uniformly at random from $H(S(\mathbf{v}))$. To show (5) we proceed by induction on the tree height.

- $h=0$:

$$\begin{aligned}\mathbb{E}\left(C_{\mathrm{SE}}\left(T_{\mathbf{v}}\right)\right)&=\mathbb{E}\left(\frac{c(\mathbf{v})}{|\mathbf{v}|}+\frac{|S(\mathbf{v})|}{|\mathbf{v}|}\cdot 0\right)\\&=\frac{c(\mathbf{v})}{|\mathbf{v}|}=\frac{\sum_{v\in\mathbf{v}}c(v)}{|\mathbf{v}|}=\frac{\mathrm{Cost}(T_{\mathbf{v}})}{|\mathbf{v}|}.\end{aligned}$$

- Suppose that the proposition is correct for heights $0,\ldots,h-1$. Combining this with (6) we get

$$\begin{aligned}\mathbb{E}\left(C_{\mathrm{SE}}\left(T_{\mathbf{v}}\right)\right)&=\mathbb{E}\left(\frac{c(\mathbf{v})}{|\mathbf{v}|}+\frac{|S(\mathbf{v})|}{|\mathbf{v}|}C_{\mathrm{SE}}\left(T_{\mathbf{W}}\right)\right)\\&=\frac{c(\mathbf{v})}{|\mathbf{v}|}+\frac{|S(\mathbf{v})|}{|\mathbf{v}|}\left(\frac{1}{d}\sum_{1\leqslant j\leqslant d}C_{\mathrm{SE}}\left(T_{\mathbf{w}_j}\right)\right)\\&\underbrace{=}_{\text{hypothesis}}\frac{c(\mathbf{v})}{|\mathbf{v}|}+\frac{|S(\mathbf{v})|}{|\mathbf{v}|}\left(\frac{1}{d}\sum_{1\leqslant j\leqslant d}\frac{\mathrm{Cost}\left(T_{\mathbf{w}_j}\right)}{|\mathbf{w}_j|}\right).\end{aligned}$$

Consider now the following two cases.

1) $|S(\mathbf{v})|\leqslant B$. Hence, $H(S(\mathbf{v}))=\{\mathbf{w}_1\}$, $|S(\mathbf{v})|=|\mathbf{w}_1|$, and $d=1$, so that

$$\begin{aligned}&\frac{c(\mathbf{v})}{|\mathbf{v}|}+\frac{|S(\mathbf{v})|}{|\mathbf{v}|}\left(\frac{1}{d}\sum_{1\leqslant j\leqslant d}\frac{\mathrm{Cost}\left(T_{\mathbf{w}_j}\right)}{|\mathbf{w}_j|}\right)\\&=\frac{c(\mathbf{v})}{|\mathbf{v}|}+\frac{|S(\mathbf{v})|}{|\mathbf{v}|}\frac{\mathrm{Cost}(T_{\mathbf{w}_1})}{|\mathbf{w}_1|}\\&=\frac{c(\mathbf{v})+\mathrm{Cost}(T_{\mathbf{w}_1})}{|\mathbf{v}|}=\frac{\mathrm{Cost}(T_{\mathbf{v}})}{|\mathbf{v}|}.\end{aligned}$$

2) $|S(\mathbf{v})|>B$. In this case, there is a set of possible hyper nodes that will be chosen uniformly at random from $H(S(\mathbf{v}))$. So, $H(S(\mathbf{v}))=\{\mathbf{w}_1,\ldots,\mathbf{w}_d\}$, $d=|H(S(\mathbf{v}))|=\binom{|S(\mathbf{v})|}{B}>1$ and $|\mathbf{w}_j|=B$ for all $j=1,\ldots,d$. We continue with

$$\begin{aligned}&\frac{c(\mathbf{v})}{|\mathbf{v}|}+\frac{|S(\mathbf{v})|}{|\mathbf{v}|}\left(\frac{1}{d}\sum_{1\leqslant j\leqslant 1}\frac{\mathrm{Cost}\left(T_{\mathbf{w}_j}\right)}{|\mathbf{w}_j|}\right)=\frac{c(\mathbf{v})}{|\mathbf{v}|}\\&+\frac{|S(\mathbf{v})|}{|\mathbf{v}|}\left(\binom{|S(\mathbf{v})|}{B}^{-1}\sum_{1\leqslant j\leqslant d}\frac{\mathrm{Cost}\left(T_{\mathbf{w}_j}\right)}{B}\right)\\&\underbrace{=}_{(3)}\frac{c(\mathbf{v})}{|\mathbf{v}|}+\frac{|S(\mathbf{v})|}{|\mathbf{v}|}\left(\binom{|S(\mathbf{v})|}{B}^{-1}\right.\\&\qquad\left.\sum_{1\leqslant j\leqslant d}\frac{\sum_{w\in\mathbf{w}_j}\mathrm{Cost}(T_w)}{B}\right)\\&\underbrace{=}_{\text{Lemma }(A.1)}\frac{c(\mathbf{v})}{|\mathbf{v}|}+\frac{|S(\mathbf{v})|}{|\mathbf{v}|}\left(\binom{|S(\mathbf{v})|}{B}^{-1}\right.\\&\qquad\left.\frac{\binom{|S(\mathbf{v})|}{B}}{|S(\mathbf{v})|}\sum_{w\in S(\mathbf{v})}\mathrm{Cost}(T_w)\right)\\&=\frac{c(\mathbf{v})}{|\mathbf{v}|}+\frac{|S(\mathbf{v})|}{|\mathbf{v}|}\left(\frac{1}{|S(\mathbf{v})|}\sum_{w\in S(\mathbf{v})}\mathrm{Cost}(T_w)\right)\\&\underbrace{=}_{(3)}\frac{c(\mathbf{v})}{|\mathbf{v}|}+\frac{|S(\mathbf{v})|}{|\mathbf{v}|}\left(\frac{\mathrm{Cost}\left(T_{S(\mathbf{v})}\right)}{|S(\mathbf{v})|}\right)\\&=\frac{c(\mathbf{v})+\mathrm{Cost}\left(T_{S(\mathbf{v})}\right)}{|\mathbf{v}|}=\frac{\mathrm{Cost}(T_{\mathbf{v}})}{|\mathbf{v}|}.\end{aligned}$$

□

If for the original tree T with root v_0 we define $\mathbf{v}_0=\{v_0\}$, then the forest $T_{\mathbf{v}_0}$ is identical to T and, with $|\mathbf{v}_0|=1$, Theorem A.1 yields the following corollary.

Corollary A.1 (Unbiased tree estimator): Let T be a tree rooted at v_0, and let $\mathbf{v}_0=\{v_0\}$. Then, SE returns unbiased estimator for the total tree cost; that is,

$$\mathbb{E}\left(C_{\mathrm{SE}}(T_{\mathbf{v}_0})\right)=\mathrm{Cost}\left(T\right).$$

□

APPENDIX B
SE EXAMPLE

We consider an example run of the SE Algorithm 3.1 while using different budget sizes. The SE algorithm does not operate on the original tree but rather on the associate "hyper tree". This crucial property (that we call *tree folding*) may lead to significant variance reduction. One should also note that the corresponding "tree folding" property is a direct consequence of the built–in splitting mechanism of the SE, see the discussion in Section III-A.

Consider a tree T displayed in Figure 8. Suppose that we wish to count the number of nodes, so $c(v)=1$ for all $v\in T$.

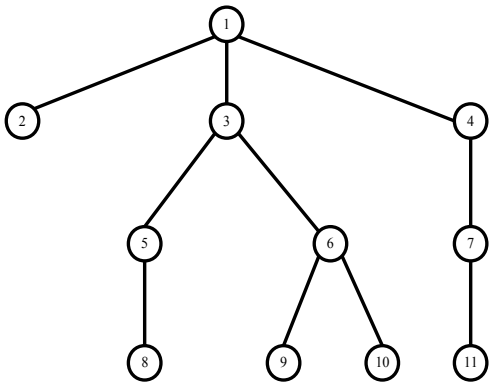

Fig. 8. A tree with 11 nodes.

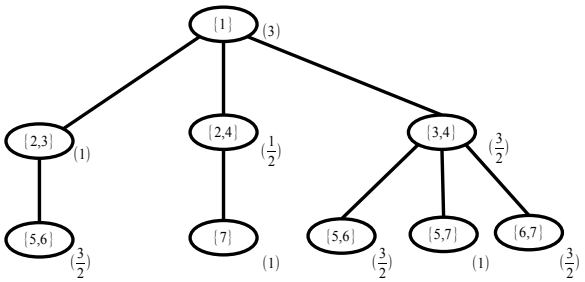

Fig. 9. SE folded tree for $B = 2$.

1) For $B = 1$, each run of Algorithm 3.1 will choose one of possible 5 paths. Below, we consider all the paths, the obtained estimator and the corresponding probabilities.
   a) $1-2$, $C_{\rm SE} = 1+3 = 4$, with probability $1/3$.
   b) $1-4-7-11$, $C_{\rm SE} = 1+3+3+3 = 10$, with probability $1/3$.
   c) $1-3-5-8$, $C_{\rm SE} = 1+3+6+6 = 16$,, with probability $1/3 \cdot 1/2 = 1/6$.
   d) $1-3-6-9$, $C_{\rm SE} = 1+3+6+12 = 22$, with probability $1/3 \cdot 1/2 \cdot 1/2 = 1/12$.
   e) $1-3-6-10$, $C_{\rm SE} = 1+3+6+12 = 22$, with probability $1/3 \cdot 1/2 \cdot 1/2 = 1/12$.

   It is easy to check that

$$\begin{aligned}\mathbb{E}\left(C_{\rm SE}\right) &= 4 \cdot 1/3 + 10 \cdot 1/3 + 16 \cdot 1/6 \\ &\quad + 22 \cdot 1/12 + 22 \cdot 1/12 = 11,\end{aligned}$$

   and that

$$\begin{aligned}\mathbb{E}\left(C_{\rm SE}^2\right) &= 4^2 \cdot 1/3 + 10^2 \cdot 1/3 + 16^2 \cdot 1/6 + 22^2 \cdot 1/12 \\ &\quad + 22^2 \cdot 1/12 = 162 \Rightarrow\end{aligned}$$

$$\begin{aligned}&\Rightarrow \mathrm{Var}\left(C_{\rm SE}\right) = \mathbb{E}\left(C_{\rm SE}^2\right) - \mathbb{E}\left(C_{\rm SE}\right)^2 \\ &= 162 - 11^2 = 41.\end{aligned}$$

2) For $B = 2$ the SE algorithm starts at the root and finds that there are 3 child nodes, see Figure 9. Recall that $B = 2$, so, we set $C_{\rm SE} \leftarrow 1$ and $D \leftarrow 3$ and continue to one of the child hyper nodes ($\{2,3\}, \{2,4\}$ or $\{2,3\}$). The process continues until reaching the leaves. Note that near each hyper node $\mathbf{X}$ we write its average degree, $|S(\mathbf{X})|/|\mathbf{X}|$, inside the brackets. For example, the hyper node $\{1\}$ has an average degree of $3/1$ and the hyper node $\{2,4\}$ has an average degree of $1/2$, because in the original tree the degree of node 2 is zero and the degree of node 4 is 1, respectively. The folded tree in Figure 9 corresponds to the SE algorithm with $B = 2$.

   Each run of Algorithm 3.1 will also choose one of possible 5 paths. Below, we consider all the paths, the obtained estimator and the corresponding probabilities.
   a) $\{1\}-\{2,3\}-\{5,6\}$, $C_{\rm SE} = 1+3+3\cdot 1+3\cdot 3/2 = 11.5$, with probability $1/3$.
   b) $\{1\}-\{2,4\}-\{7\}$, $C_{\rm SE} = 1+3+3\cdot 1/2+3/2\cdot 1 = 7$, with probability $1/3$.
   c) $\{1\}-\{3,4\}-\{5,6\}$, $C_{\rm SE} = 1+3+3\cdot 3/2+9/2\cdot 3/2 = 15.25$, with probability $1/9$.
   d) $\{1\}-\{3,4\}-\{5,7\}$, $C_{\rm SE} = 1+3+3\cdot 3/2+9/2\cdot 1 = 13$, with probability $1/9$.
   e) $\{1\}-\{3,4\}-\{6,7\}$, $C_{\rm SE} = 1+3+3\cdot 3/2+9/2\cdot 3/2 = 15.25$, with probability $1/9$.

   It is easy to check that

$$\begin{aligned}\mathbb{E}\left(C_{\rm SE}\right) &= 11.5 \cdot 1/3 + 7 \cdot 1/3 + 15.25 \cdot 2/9 \\ &\quad + 22 \cdot 1/12 + 13 \cdot 1/9 = 11,\end{aligned}$$

   and that

$$\begin{aligned}\mathbb{E}\left(C_{\rm SE}^2\right) &= 11.5^2 \cdot 1/3 + 7^2 \cdot 1/3 + 15.25^2 \cdot 2/9 + 22 \cdot 1/12 \\ &\quad + 13^2 \cdot 1/9 = 130.875 \Rightarrow\end{aligned}$$

$$\begin{aligned}&\Rightarrow \mathrm{Var}\left(C_{\rm SE}\right) = \mathbb{E}\left(C_{\rm SE}^2\right) - \mathbb{E}\left(C_{\rm SE}\right)^2 \\ &= 130.875 - 11^2 = 9.875.\end{aligned}$$

   This example illustrates the power of the SE *tree folding*. Recall that the variance for the $B = 1$ case was equal to 41, but, despite of the fact that the budget of SE is only twice as compared to the previous case, (1), the SE estimator has a variance of 9.875, hence, we obtained factor 4 variance reduction.
3) For $B = 3$ there is no tree level such that the number of nodes in it is greater than B, so the original tree collapses to a trivial hyper tree where all the hyper node degrees are equal to 1. This will immediately result in a zero variance estimator. Consider the folded tree in Figure 10 that corresponds to the SE algorithm with $B = 3$.

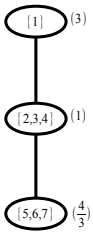

Fig. 10. SE tree for $B = 3$.

Algorithm 3.1 will find a single path $\{1\} - \{2,3,4\} - \{5,6,7\}$, (with probability 1), such that

$$C_{\mathrm{SE}} = 1 + 3 \cdot 1 + 3 + 3 \cdot 1 + 3 \cdot 4/3 = 11$$

which results in a zero variance estimator.

REFERENCES

[1] D. P. Kroese, T. Taimre, and Z. I. Botev, *Handbook of Monte Carlo methods*. New York: John Wiley & Sons, 2011.

[2] I. B. Gertsbakh and Y. Shpungin, *Models of Network Reliability: Analysis, Combinatorics, and Monte Carlo*. Boca Raton, FL, USA: CRC Press, Inc., 2009.

[3] D. R. Karger, "A randomized fully polynomial time approximation scheme for the all terminal network reliability problem," in *SIAM Journal on Computing*, 1996, pp. 11–17.

[4] L. G. Valiant, "The complexity of enumeration and reliability problems," *SIAM Journal on Computing*, vol. 8, no. 3, pp. 410–421, 1979.

[5] R. M. Karp and M. Luby, "Monte-Carlo algorithms for enumeration and reliability problems," in *Proceedings of the 24th Annual Symposium on Foundations of Computer Science*, 1983, pp. 56–64.

[6] M. Dyer, "Approximate counting by dynamic programming," in *Proceedings of the 35th ACM Symposium on Theory of Computing*, 2003, pp. 693–699.

[7] M. Jerrum, A. Sinclair, and E. Vigoda, "A polynomial-time approximation algorithm for the permanent of a matrix with non-negative entries," *Journal of the ACM*, pp. 671–697, 2004.

[8] M. Dyer, A. Frieze, and M. Jerrum, "On counting independent sets in sparse graphs," in *In 40th Annual Symposium on Foundations of Computer Science*, 1999, pp. 210–217.

[9] S. P. Vadhan, "The complexity of counting in sparse, regular, and planar graphs," *SIAM Journal on Computing*, vol. 31, pp. 398–427, 1997.

[10] M. Jerrum, L. G. Valiant, and V. V. Vazirani, "Random generation of combinatorial structures from a uniform distribution," *Theoretical Computer Science*, vol. 43, pp. 169–188, 1986.

[11] Z. I. Botev and D. P. Kroese, "Efficient Monte Carlo simulation via the generalized splitting method," *Statistics and Computing*, vol. 22, pp. 1–16, 2012.

[12] M. Jerrum and A. Sinclair, "The Markov chain Monte Carlo method: An approach to approximate counting and integration," in *Approximation Algorithms for NP-hard Problems*, D. Hochbaum, Ed. PWS Publishing, 1996, pp. 482–520.

[13] R. Y. Rubinstein, "The Gibbs cloner for combinatorial optimization, counting and sampling," *Methodology and Computing in Applied Probability*, vol. 11, pp. 491–549, 2009.

[14] R. Y. Rubinstein, A. Dolgin, and R. Vaisman, "The splitting method for decision making," *Communications in Statistics - Simulation and Computation*, vol. 41, no. 6, pp. 905–921, 2012.

[15] Z. I. Botev, P. L'Ecuyer, G. Rubino, R. Simard, and B. Tuffin, "Static network reliability estimation via generalized splitting," *INFORMS Journal on Computing*, vol. 25, no. 1, pp. 56–71, Jan. 2013.

[16] J. K. Blitzstein and P. Diaconis, "A sequential importance sampling algorithm for generating random graphs with prescribed degrees," *Internet Mathematics*, vol. 6, no. 4, pp. 489–522, 2011.

[17] Y. Chen, P. Diaconis, S. P. Holmes, and J. S. Liu, "Sequential Monte Carlo methods for statistical analysis of tables," *Journal of the American Statistical Association*, vol. 100, pp. 109–120, March 2005.

[18] R. Y. Rubinstein, "Stochastic Enumeration Method for counting NP-hard problems," *Methodology and Computing in Applied Probability*, pp. 1–42, 2012.

[19] J. Blanchet and D. Rudoy, "Rare event simulation and counting problems," in *Rare Event Simulation Using Monte Carlo Methods*, 1st ed. New York: Wiley, 2009.

[20] P. L'Ecuyer, G. Rubino, S. Saggadi, and B. Tuffin, "Networks – approximate zero-variance importance sampling for static network reliability estimation," *IEEE transactions on reliability*, 2011.

[21] 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, 1991.

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

[23] ——, *Network Reliability and Resilience*, ser. SpringerBriefs in Electrical and Computer Engineering. Springer, 2011.

[24] I. B. Gertsbakh, Y. Shpungin, and R. Vaisman, "Network reliability Monte Carlo with nodes subject to failure," *International Journal on Performability Engineering*, vol. 10, no. 2, pp. 163 – 172, 2014.

[25] F. J. Samaniego, *System Signatures and Their Applications in Engineering Reliability*, ser. International Series in Operations Research & Management Science. Springer, 2007.

[26] D. E. Knuth, "Estimating the efficiency of backtrack programs," *Math. Comp.*, vol. 29, 1975.

[27] R. Vaisman, "Stochastic enumeration methods for counting, rare-events and optimization," Ph.D. dissertation, Technion, Israel Institute of Technology, Technion, Haifa, Israel, 32000, October 2013.

[28] R. Y. Rubinstein, A. Ridder, and R. Vaisman, *Fast Sequential Monte Carlo Methods for Counting and Optimization*. New York: John Wiley and Sons, 2013.

[29] M. J. J. Garvels, "The splitting method in rare event simulation," Ph.D. dissertation, Enschede, October 2000.

[30] P. Glasserman, P. Heidelberger, P. Shahabuddin, and T. Zajic, "Splitting for rare event simulation: Analysis of simple cases." in *Winter Simulation Conference*, 1996, pp. 302–308.

[31] F. Harary, J. P. Hayes, and H.-J. Wu, "A survey of the theory of hypercube graphs," *Computers & Mathematics with Applications*, vol. 15, no. 4, pp. 277 – 289, 1988.

Radislav Vaisman is a postdoctoral research fellow in the School of Mathematics and Physics at the University of Queensland. He received the PhD degree in Information System Engineering from the Technion, Israel Institute of Technology. His research interests are rare–event probability estimation, theoretical computer science and randomized algorithms. He is the co-author of two books, *Fast Sequential Monte Carlo Methods for Counting and Optimization* and *Ternary Networks: Reliability and Monte Carlo*. His personal website can be found under http://www.smp.uq.edu.au/node/106/2407. His email address is r.vaisman@uq.edu.au.

Dirk P. Kroese is a Professor of Mathematics and Statistics in the School of Mathematics and Physics at the University of Queensland. He is the co-author of several influential monographs on simulation and Monte Carlo methods, including *Handbook of Monte Carlo Methods* and *Simulation and the Monte Carlo Method (2nd Edition)*. Dirk is a pioneer of the well-known Cross-Entropy method — an adaptive Monte Carlo technique, invented by Reuven Rubinstein, which is being used around the world to help solve difficult estimation and optimization problems in science, engineering, and finance. His personal website can be found under http://www.maths.uq.edu.au/~kroese. His email address is kroese@maths.uq.edu.au.

Ilya B. Gertsbakh is Professor Emeritus in the Department of Mathematics at Ben Gurion University of the Negev. He is author or co-author of seven books, one of which is *Models of Network Reliability: Analysis, Combinatorics and Monte Carlo* (coauthored with Yoseph Shpungin). Ilya Gertsbakh has published some 100 papers on topics of operations research, reliability, applied probability, and Monte Carlo. He received his MSc degrees in Mechanical Engineering and Mathematics (1961) from Latvian State University, Riga, and his PhD degree (1964) in applied probability and statistics from the Latvian Academy of Sciences, also in Riga.