Probability in the Engineering and Informational Sciences, **17**, 2003, 23–45. Printed in the U.S.A.

THE TREE CUT AND MERGE ALGORITHM FOR ESTIMATION OF NETWORK RELIABILITY

K.-P. Hui
Information Networks Division
DSTO
Edinburgh 5111, Australia
E-mail: Kin-ping.Hui@dsto.defence.gov.au

N. Bean
Department of Applied Mathematics
University of Adelaide
Adelaide 5005, Australia
E-mail: nbean@maths.adelaide.edu.au

M. Kraetzl
ISR Division
DSTO
Edinburgh 5111, Australia
E-mail: Miro.Kraetzl@dsto.defence.gov.au

D. Kroese
Department of Mathematics
University of Queensland
Brisbane 4072, Australia
E-mail: kroese@maths.uq.edu.au

This article presents Monte Carlo techniques for estimating network reliability. For highly reliable networks, techniques based on graph evolution models provide very good performance. However, they are known to have significant simulation cost. An existing hybrid scheme (based on partitioning the time space) is available to speed up the simulations; however, there are difficulties with optimizing the important parameter associated with this scheme. To overcome these difficulties, a new hybrid scheme (based on partitioning the edge set) is proposed in this article. The proposed scheme shows orders of magnitude improvement of performance over the existing techniques in certain classes of network. It also provides reliability bounds with little overhead.

 0269-9648/03 $12.50

1. INTRODUCTION

1.1. Problem Description

It is well known that, for large networks, the exact calculation of network reliability is difficult. Indeed, computing the probability that a graph is connected is a #P-complete problem [2,11]. Hence, for large networks, estimating the reliability using simulation techniques becomes necessary. In highly reliable networks such as modern communication networks, the probability of network failure is very low. Direct simulation of such rare events is slow and, hence, very expensive. Various techniques have been developed to produce better estimates. For example, Kumamoto proposed a very simple technique called *Dagger Sampling* to improve the Crude Monte Carlo simulation [8]. Fishman proposed *Procedure Q*, which can provide reliability estimates as well as bounds [7]. Colbourn and Harms proposed a technique that will provide progressive bounds that will eventually converge to an exact reliability value [3]. Easton and Wong proposed a sequential construction method [4]. Elperin, Gertsbakh, and Lomonosov proposed *Evolution Models* for estimating reliability of highly reliable networks [5,6,9,10].

For highly reliable networks such as modern communication networks, the *Merge Process* proposed by Elperin et al. provides good performance without significant overhead [5]. However, it has a relatively high computation cost per sample. To combat the high complexity for the Merge Process, Lomonosov proposed a hybrid scheme based on partitioning the time space [9]. This scheme can reduce the average simulation cost; however, the choice of a partition point in time space controls the performance of the scheme. Unfortunately, the optimal value for this parameter is difficult to find. In this article, we develop a new hybrid scheme for the Merge Process by using a novel partitioning based on the edge set. In certain classes of networks, the new scheme shows orders of magnitude improvement in performance. In addition to performance improvement, the new scheme can also provide reliability bounds, which has, thus far, never been available using Evolution Models.

In this article, we first define network reliability in Section 1.2. Then, the mathematical formulation of the Evolution Models is reviewed in Section 2. The existing hybrid scheme is presented in Section 3 using our mathematical formulation, followed by the proposed new scheme in Section 4. Experiments and results are presented in Section 5 with some discussions.

1.2. Network Reliability

Consider an undirected graph (or network) $\mathcal{G}(V, E, K)$, where V is the set of n vertices (or nodes), E is the set of m edges, and $K \subseteq V$ is a set of *terminal* nodes, with $|K| \geq 2$. Associated with each edge $e \in E$ is a binary random variable X_e, denoting the *failure state* of the edge. In particular, $\{X_e = 1\}$ is the event that the edge is operational, and $\{X_e = 0\}$ is the event that it has failed. We label the edges from 1 to m and call the vector $X = (X_1, \ldots, X_m)$ the (failure) state of the network, or the state of the set E. Let $\mathcal{S}$ be the set of all 2^m possible states of E.

Notation 1: For $A \subseteq E$, let $x = (x_1, \ldots, x_m)$ be the vector in $\{0,1\}^m$ with

$$x_i = \begin{cases} 1, & i \in A \\ 0, & i \notin A. \end{cases}$$

We can *identify* x with the set A. Henceforth, we will use this identification whenever this is convenient.

Next, we assume that the random variables $\{X_e,\ e \in E\}$ are mutually independent. Let p_e and q_e denote the reliability and unreliability of $e \in E$, respectively; that is,

$$p_e = \mathbb{P}[X_e = 1],$$
$$q_e = \mathbb{P}[X_e = 0] = 1 - p_e.$$

The reliability $r(\mathcal{G}; p)$ of the network is defined as the probability of K being *connected* by operational edges. Further, let $p = (p_1, \ldots, p_m)$. Thus,

$$r(\mathcal{G}; p) = \mathbb{E}[\varphi(X)] = \sum_{x \in \mathcal{S}} \varphi(x)\mathbb{P}[X = x], \tag{1}$$

where

$$\varphi(x) = \begin{cases} 1 & \text{if } K \text{ is connected} \\ 0 & \text{otherwise.} \end{cases}$$

This is the standard formulation of the reliability of unreliable systems (networks); see, for example, [1]. The function φ is called the *structure function* of the unreliable system. Note that the reliability of the network is completely determined by the individual edge reliabilities since we do not consider node failures.

In the rest of this article, when $\mathcal{G}$ and p are assumed to be understood, we will write r instead of $r(\mathcal{G}; p)$. For highly reliable networks, it is sometimes more useful to analyze or estimate the system unreliability

$$\bar{r} = 1 - r.$$

1.3. Crude Monte Carlo Simulation

The easiest way to estimate r (or $\bar{r}$) is to use Crude Monte Carlo (CMC) simulation. Let $X^{(1)}, \ldots, X^{(N)}$ be independent and identically distributed (i.i.d.) random vectors with the same distribution as X. Then,

$$\hat{r} = \frac{1}{N} \sum_{i=1}^{N} \varphi(X^{(i)})$$

is an unbiased estimator for r. An important measure for the "efficiency" of any estimator is its *relative error*. The relative error for $\hat{r}$ is given by

$$\mathrm{re}(\hat{r}) = \sqrt{\frac{\mathrm{Var}(\hat{r})}{(\mathbb{E}[\hat{r}])^2}} = \sqrt{\frac{r(1-r)/N}{r^2}} = \sqrt{\frac{1-r}{Nr}}.$$

Similarly, the relative error for $\hat{\bar{r}}$ is

$$\mathrm{re}(\hat{\bar{r}}) = \sqrt{\frac{1-\bar{r}}{N\bar{r}}}.$$

This shows that for small $\bar{r}$ (which is typical in communication networks), a large sample size is needed to estimate $\bar{r}$ accurately. When $\bar{r}$ is small, the event that the terminal nodes are not connected is a *rare event*. In the next section, we discuss methods to increase the accuracy of simulation procedures, which work well for rare events.

2. PERMUTATION MONTE CARLO SIMULATION

2.1. Construction Process

First, we describe the concepts behind the *Permutation Monte Carlo* (π-MC) simulation (see [5] for the original description). Consider the network $\mathcal{G}(V,E)$ in which each edge e has an exponential repair time with repair rate $\lambda(e) = -\log(q_e)$. At time $t = 0$, all edges are failed. We assume that all repair times are independent of each other. The state of e at time t is denoted by $X_e(t)$ and the state of the edge set E at time t is given by the vector $X(t)$, defined in a similar way as earlier. Thus, $(X(t))$ is a Markov process with state space $\{0,1\}^m$ or, in view of Notation 1, a Markov process on subsets of E. This process is called the *Construction Process* (CP) of the network.

Let Π denote the *order* in which the edges are constructed (become operational), and let $S_0, S_0 + S_1, \ldots, S_0 + \cdots + S_{m-1}$ be the times at which those edges are constructed. Hence, the S_i are *sojourn times* of $(X(t))$. Π is a random variable which takes values in the space of permutations of E.

For any permutation $\pi = (e_1, e_2, \ldots, e_m)$, define

$$\begin{aligned} E_0 &= E, \\ E_i &= E_{i-1} \backslash \{e_i\}, \qquad 1 \le i \le m-1, \\ \lambda(E_i) &= \sum_{e \in E_i} \lambda(e), \end{aligned}$$

and let

$$b(\pi) = \min_i \{\varphi(E_i) = 1\}$$

be the *critical number* of π.

From the general theory of Markov processes, it is not difficult to see that

$$\mathbb{P}[\Pi = \pi] = \prod_{j=1}^{m} \frac{\lambda(e_j)}{\lambda(E_{j-1})}.$$

Moreover, conditional on $\{\Pi = \pi\}$, the sojourn times $S_0, \ldots, S_{m-1}$ are independent and each S_i is exponentially distributed with parameter $\lambda(E_i)$, $i = 0, \ldots, m-1$.

Note that the probability of each edge e being operational at time $t = 1$ is p_e. It follows that the *network* reliability at time $t = 1$ is the same as in (1). Hence, by conditioning on Π, we have

$$r = \mathbb{E}[\varphi(X(1))] = \sum_{\pi} \mathbb{P}[\Pi = \pi]\mathbb{P}[\varphi(X(1)) = 1 | \Pi = \pi], \tag{2}$$

or

$$\bar{r} = 1 - r = \sum_{\pi} \mathbb{P}[\Pi = \pi]\mathbb{P}[\varphi(X(1)) = 0 | \Pi = \pi]. \tag{3}$$

Using the definitions of S_i and $b(\pi)$, we can write the last probability in terms of convolutions of exponential distribution functions; namely for any $t \geq 0$, we have

$$\begin{aligned} \mathbb{P}[\varphi(X(t)) = 0 | \Pi = \pi] &= \mathbb{P}[S_0 + \cdots + S_{b(\pi)-1} > t | \Pi = \pi] \\ &= 1 - \mathrm{Conv}_{0 \leq i < b(\pi)}\{1 - \exp[-\lambda(E_i)t]\}. \end{aligned} \tag{4}$$

Let

$$g_C(\pi) = \mathbb{P}[\varphi(X(1)) = 0 | \Pi = \pi],$$

as given in (4). Equation (3) can be rewritten as

$$\bar{r} = \mathbb{E}[g_C(\Pi)],$$

and this shows how the Permutation Monte Carlo simulation scheme works; namely let $\Pi^{(1)}, \ldots, \Pi^{(N)}$ be i.i.d. random permutations, each distributed according to Π. Then,

$$\hat{\bar{r}} = \frac{1}{N} \sum_{i=1}^{N} g_C(\Pi^{(i)})$$

is an unbiased estimator for $\bar{r}$.

2.1.1. Performance of the Construction Process. The Construction Process scheme based on generating permutations π and computing $g_C(\pi)$ is characterized by the variance:

$$\mathrm{Var}_{\mathrm{CP}} = \sum_{\pi} \mathbb{P}[\Pi = \pi] g_C^2(\pi) - \left(\sum_{\pi} \mathbb{P}[\Pi = \pi] g_C(\pi) \right)^2. \tag{5}$$

Comparing this with the variance of the Crude Monte Carlo scheme,

$$\mathrm{Var}_{\mathrm{CMC}} = r(1-r) = \left(\sum_{\pi} \mathbb{P}[\Pi = \pi] g_C(\pi)\right)\left(1 - \sum_{\pi} \mathbb{P}[\Pi = \pi] g_C(\pi)\right)$$

$$= \mathrm{Var}_{\mathrm{CP}} + \sum_{\pi} \mathbb{P}[\Pi = \pi] g_C(\pi)(1 - g_C(\pi)),$$

proved that the Construction Process scheme has a lower variance than the Crude Monte Carlo scheme and hence greater accuracy. However, this accuracy comes at the expense of more complex computations.

2.2. Standard Merge Process

A closer look at the evolution of the Construction Process process reveals that many of the above results remain valid when we *merge* various states into "superstates" at various stages of the process. This is known as the *Merge Process*. We will briefly describe the ideas below (see [9] for a detailed description).

To begin with, for a subset $F \subseteq E$ of edges, denote by $\sigma = \{V_1, V_2, \ldots, V_k\}$ the *proper partition* (in connected components) of the subgraph $\mathcal{G}(V, F)$ (including isolated nodes, if any). Let I_i denote the edge set of the induced subgraph $\mathcal{G}(V_i)$. The set $I_\sigma = I_1 \cup \cdots \cup I_k$ of *inner* edges (i.e., the edges within the components) is the closure of F (denoted by $\langle F \rangle$). Denote its complement (the *intercomponent* edges) by $E_\sigma = E \setminus I_\sigma$. Figure 1 is an example of a complete six-node graph (K_6), a subgraph, and its corresponding closure.

Let $\mathbb{L}(\mathcal{G})$ be the collection of all proper partitions of $\mathcal{G}(V, E)$. It is not difficult to see that $\mathbb{L}(\mathcal{G})$ is a *lattice*, ordered by the relation $\tau \prec \sigma \Leftrightarrow I_\tau \subset I_\sigma$ (i.e., σ is obtained by merging components of τ).

Consider the Construction Process $(X(t))$ of the network. By restricting the process $(X(t))$ to $\mathbb{L}(\mathcal{G})$, we obtain another Markov process $(\mathbb{X}(t))$, called the *Merge Process* (MP) of the network. This process starts from the initial state σ_0 of isolated nodes and ends at the terminal state σ_ω corresponding to $\mathcal{G}(V, E)$.

Figure 2 shows $\mathbb{L}(K_4)$, the lattice of all regular partitions of the complete four-node graph K_4, grouped into four different levels according to the number of

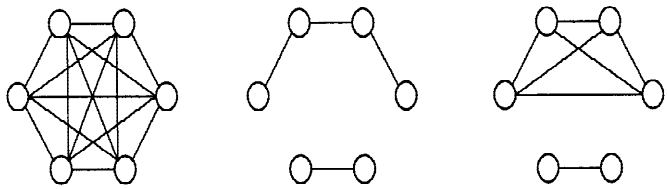

FIGURE 1. K_6, a subgraph, and its corresponding closure.

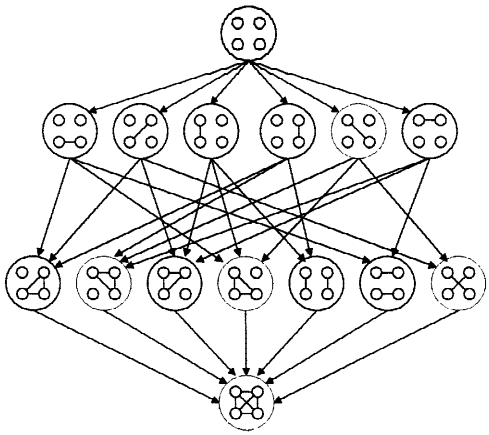

FIGURE 2. State transition diagram for merge process of K_4.

components. The arrows show the direct successions in $\mathbb{L}(K_4)$, thus forming the transition graph of the Markov process $(\mathbb{X}(t))$.

For each $\sigma \in \mathbb{L}(\mathcal{G})$, the sojourn time in σ has an exponential distribution with parameter $\lambda(\sigma) = \sum_{e \in E_\sigma} \lambda(e)$, independent of everything else. Moreover, the transition probability from τ to σ (where σ is a direct successor of τ) is given by

$$\frac{\lambda(\tau) - \lambda(\sigma)}{\lambda(\tau)}.$$

Next, in analogy with the results for the Construction Process, we define a *trajectory* of $(\mathbb{X}(t))$ as a sequence $\theta = (\sigma_0, \sigma_1, \ldots, \sigma_b)$, where $b = b(\theta)$ is the first index i such that σ_i is "up"; that is, the network is operational. By defining $\varphi(\mathbb{X}(t)) = \varphi(X(t))$, we have

$$\bar{r} = \mathbb{P}[\varphi(\mathbb{X}(1)) = 0] = \mathbb{E}[g_M(\Theta)],$$

where Θ is the random trajectory of $(\mathbb{X}(t))$. For each outcome $\theta = (\sigma_0, \ldots, \sigma_b)$ of Θ, $g_M(\theta)$ is given by

$$g_M(\theta) = \mathbb{P}[\varphi(\mathbb{X}(1)) = 0 | \Theta = \theta] = \mathbb{P}[S_0 + \cdots + S_{b(\theta)-1} > 1 | \Theta = \theta],$$

where S_i is the sojourn time at σ_i. Therefore, $g_M(\theta)$ is given by the value

$$1 - \mathrm{Conv}_{0 \le i < b(\theta)}\{1 - \exp[-\lambda(\sigma_i)t]\},$$

at $t = 1$.

2.2.1. Performance of the Merge Process. The Merge Process scheme based on generating permutations θ and computing $g_M(\theta)$ is characterized by the variance

$$\mathrm{Var}_{\mathrm{MP}} = \left(\sum_{\theta} \mathbb{P}[\Theta = \theta] g_M^2(\theta) \right) - \left(\sum_{\theta} \mathbb{P}[\Theta = \theta] g_M(\theta) \right)^2. \tag{6}$$

By expanding the expression for the variance of the Crude Monte Carlo, we get

$$\begin{aligned} \mathrm{Var}_{\mathrm{CMC}} &= r(1-r) \\ &= \mathrm{Var}_{\mathrm{MP}} + \sum_{\theta} \mathbb{P}[\Theta = \theta] g_M(\theta)(1 - g_M(\theta)), \end{aligned} \tag{7}$$

which shows that the Merge Process has a smaller variance than does the Crude Monte Carlo.

To compare the Merge Process with the Construction Process, we go back to the permutation $\pi = (e_1, e_2, \ldots)$ of the Construction Process, which produces a unique trajectory of the Merge Process denoted by $\langle \pi \rangle$. We say that permutations π_1, π_2 of the Construction Process are equivalent in the Merge Process if $\langle \pi_1 \rangle = \langle \pi_2 \rangle$. Thus, a trajectory θ of the Merge Process represents the set of permutations of the Construction Process satisfying $\langle \pi \rangle = \theta$; we write it as $\pi \in \theta$. For a given trajectory θ and its corresponding permutations $\pi \in \theta$, the probability of randomly choosing a particular permutation is given by

$$\mathbb{P}[\Pi = \pi | \Theta = \theta] = \frac{\mathbb{P}[\Pi = \pi, \Theta = \theta]}{\mathbb{P}[\Theta = \theta]},$$

and the reliability estimation functions are related by

$$g_M(\theta) = \sum_{\pi \in \theta} \mathbb{P}[\Pi = \pi | \Theta = \theta] g_C(\pi).$$

By the variance expansion, we get

$$\mathrm{Var}_{\mathrm{CP}} = \mathrm{Var}_{\mathrm{MP}} + \sum_{\theta} \mathbb{P}[\Theta = \theta] \left(\sum_{\pi \in \theta} \mathbb{P}[\Pi = \pi | \Theta = \theta] g_C^2(\pi) - g_M^2(\theta) \right). \tag{8}$$

The second term on the right-hand side of (8) is the part of $\mathrm{Var}_{\mathrm{CP}}$ eliminated by the state-space reduction when the Construction Process is transformed into the Merge Process. Because the term in brackets

$$\sum_{\pi \in \theta} \mathbb{P}[\Pi = \pi | \Theta = \theta] g_C^2(\pi) - g_M^2(\theta) = \mathrm{Var}_{\pi \in \theta}(g_C(\pi)) \geq 0, \tag{9}$$

(8) and (9) demonstrate that the Merge Process scheme has greater accuracy than the Construction Process scheme.

2.3. General Formulation

The above Construction Process and Merge Process simulations always start with σ_0, the isolated node state. This can be generalized to start at any state y and produce the hybrid sampling schemes described in Section 3. In this subsection, we present the ideas behind the Merge Process where it starts from a general state y.

Consider a continuous-time Markov process $(X(t), t \geq 0)$ on a finite poset (partially ordered set) $\mathcal{X}$ of states. From each state $x \in \mathcal{X}$, the Markov process can only jump to a direct successor of x; we say that the Markov process is *monotone* on $\mathcal{X}$. Let φ be a binary monotone function on $\mathcal{X}$; that is, $\varphi(x) \leq \varphi(y)$ whenever $x \leq y$. Using φ we partition $\mathcal{X}$ into two sets, $\mathcal{U} = \{x : \varphi(x) = 1\}$ (up states) and $\mathcal{D} = \{x : \varphi(x) = 0\}$ (down states).

Assume that the states in $\mathcal{U}$ are *absorbing*. We are interested in the probability that the Markov process, starting from some initial state Y, is in $\mathcal{U}$ at a certain time t. We denote this by $P(t)$. To avoid trivial cases, we assume that the starting state $Y \in \mathcal{D}$.

For $x \in \mathcal{D}$, let

$$\lambda(x) = \sum_{i \in x} \lambda_i$$

and assume $0 < \lambda(x) < \infty$. In other words, the sojourn time in x has an exponential distribution with parameter $\lambda(x)$. Let B be the random variable (and b its corresponding outcome) describing the first index such that the process is in $\mathcal{U}$, and let $\Theta_Y = (Y, X_1, \ldots, X_{B-1})$ be the sequence of consecutive states visited by $(X(t))$ before absorption. Denote the corresponding sojourn times by $S_0, S_1, \ldots, S_{B-1}$. Each outcome $\theta_y = (y, x_1, \ldots, x_{b-1})$ of Θ_Y is a *trajectory* of the process starting at y. Conditional on $\{\Theta_Y = \theta_y\}$, the sojourn times are independent and exponentially distributed with parameters $\lambda(y), \lambda(x_1), \ldots, \lambda(x_{b-1})$, respectively. As a result, we have

$$P(t) = \mathbb{P}[\varphi(X(t)) = 1] = \mathbb{E}[g(\Theta_Y, t)], \tag{10}$$

where for each $\theta_y = (x_0 = y, x_1, \ldots, x_{b-1})$,

$$g(\theta_y, t) = \mathbb{P}[S_0 + \cdots + S_{b-1} > t | \Theta_Y = \theta_y] = 1 - \mathrm{Conv}_{0 \leq i < b}\{1 - \exp[-\lambda(x_i)t]\}.$$

Hence, we may estimate $P(t)$ by simulating N independent copies $\Theta_Y^{(1)}, \ldots, \Theta_Y^{(N)}$ of Θ_Y and evaluating the estimator

$$\widehat{P(t)} = \frac{1}{N} \sum_{i=1}^{N} g(\Theta_Y^{(i)}, t).$$

Note that the generalization is equally applicable to any of the $g(\cdot)$ of evolution models.

3. HYBRID SAMPLING SCHEMES

The complexity of a single sample in the Merge Process, including generating a trajectory and computing the convolution, is $O(n^2)$ (see [5]) as compared to almost $O(n)$ in the Crude Monte Carlo. This complexity is acceptable as a price for the low relative error when $\bar{r} \to 0$. Lomonosov suggested a hybrid scheme (which we call the *Leap–Evolve* scheme) to reduce the average complexity when $\bar{r}$ is not so near zero [9].

The scheme is based on the following observation. Let $(X(t))$ be the Construction Process or Merge Process as described earlier, with repair rate $\lambda(e) = -\log(q_e)$, as earlier, and starting with all links down. By conditioning on the state of the system at time $s \in [0,1]$ and the trajectory Θ of $(X(t))$ after time s, we find

$$\begin{aligned}
\bar{r} &= \mathbb{P}[\varphi(X(1)) = 0] = \mathbb{P}[\varphi(X(s)) = 0, \varphi(X(1)) = 0] \\
&= \mathbb{E}[\mathbb{E}[I_{\{\varphi(X(s))=0\}} I_{\{\varphi(X(1))=0\}} | X(s), \Theta_{X(s)}]] \\
&= \mathbb{E}[I_{\{\varphi(X(s))=0\}} \mathbb{P}[\varphi(X(1)) = 0 | \Theta_{X(s)}]] \\
&= \mathbb{E}[I_{\{\varphi(X(s))=0\}} g_M(\Theta_{X(s)}, 1-s)].
\end{aligned}$$

Here, I_A denotes the indicator function of the event A. Note that for fixed θ_y, $g_M(\theta_y, 1-s)$ can be evaluated by applying the convolution given in Section 2.

Figure 3 shows an example of the Leap–Evolve scheme using the same lattice as in Figure 2. However, the Merge Process starts at time s. Depending on the outcome $X(s)$, the initial state of the Merge Process can be in any state in the lattice.

The hybrid simulation now involves partitioning the time space into two parts: $[0, s)$ and $[s, \infty)$. Each simulation run consists of two steps:

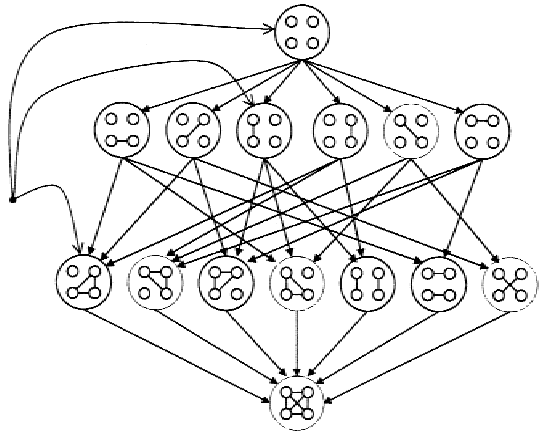

FIGURE 3. State transition diagram for the Leap–Evolve scheme of K_4.

1. *Leap*: An outcome of the random variable $X(s)$ is generated by independently bringing the edges up with probabilities $1-\exp[-\lambda(e)s]$, $e \in E$. The complexity of this step is close to $O(n)$.
2. *Evolution*: If the outcome y of $X(s)$ is in $\mathcal{U}$, let $z=0$. If the outcome y of $X(s)$ is in $\mathcal{D}$, generate a trajectory Θ_y on the interval $(s,\infty]$, starting $(X(t))$ in state y at time s. For an outcome θ_y of Θ_y, let $z = g_M(\theta_y, 1-s)$. The complexity of generating $g_M(\theta_y, 1-s)$ is $O(|y|^2)$.

Note that by the above construction, z is indeed an outcome of the random variable $I_{\{\varphi(X(s))=0\}} g_M(\Theta_{X(s)}, 1-s)$.

Hence, if $Z^{(1)}, \ldots, Z^{(N)}$ are independent samples from Z, then

$$\hat{\bar{r}} = \frac{1}{N}\sum_{i=1}^{N} Z^{(i)}$$

is an unbiased estimator of $\bar{r}$.

3.1. Performance of the Leap–Evolve Scheme

The complexity of generating $g_M(\theta_y, 1-s)$ is $O(|y|^2)$. The mean complexity of the evolution step of the hybrid scheme is, therefore, at most

$$C\sum_{y} \mathbb{P}[X(s)=y]|y|^2,$$

where C is some constant.

In the Leap–Evolve sampling scheme, the choice of s can be critical. If s is too large, $X(s)$ may converge to $\mathcal{U}$ and leave no room for evolution sampling to lower the relative error in comparison to the ordinary Crude Monte Carlo scheme. If s is too small, then $|X(s)|$ will be "close" to the initial state and will hardly reduce the average sampling complexity in comparison to the ordinary Merge Process. In Section 4, we introduce an alternative hybrid scheme that avoids this critical choice of s.

4. TREE CUT AND MERGE ALGORITHM

4.1. Cut and Merge

A different hybrid sampling scheme is proposed here. Instead of partitioning the time space, we propose partitioning the edge set into $F \subseteq E$ and its complement $\bar{F} = E \setminus F$.

Let $\mathbb{L}$ be the *lattice* of all *proper partitions* of $\mathcal{G}(V,E)$ as described in Section 3. For each $\sigma \in \mathbb{L}$ and edge set $\bar{F} \subseteq E$ we define the sublattice $\mathbb{L}_\sigma^{\bar{F}}$ of $\mathbb{L}$ as the set of all successors of σ that can be obtained by merging *only* the edges in $\bar{F}$. Figure 4 shows the sublattice induced by the edge set shown in the lower left corner of the figure. Note that, as in this example, it is possible that $\mathbb{L}_\sigma^{\bar{F}}$ does not have an "up" state.

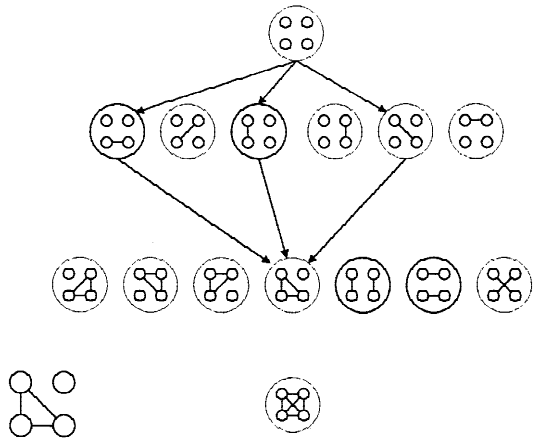

FIGURE 4. A sublattice of the K_4 graph induced by the edge set shown in the lower left corner.

Let $A \subseteq F$. Consider a Markov process $(\widetilde{X}(t))$ on E with transition rates $\lambda(e) = 0$ for $e \in F$ and with the edges $e \in A$ *operational* at time 0; the other transition rates are the same as for our original edge state process $(X(t))$ in Section 3. Let σ be the state in $\mathbb{L}$ corresponding to A. The Markov process $(\widetilde{X}(t))$ induces another Markov process, $(\widetilde{\mathbb{X}}(t))$ say, on $\mathbb{L}_{\sigma}^{\bar{F}}$, in the same way as $(X(t))$ induces $(\mathbb{X}(t))$ on $\mathbb{L}$.

Let Θ be the random trajectory of $(\widetilde{\mathbb{X}}(t))$ and Σ be the random variable describing the state in $\mathbb{L}$ corresponding to $A \subseteq F$. For each outcome σ of Σ and θ_σ of Θ_Σ, let

$$g_T(\theta_\sigma) = \mathbb{P}[\varphi(\widetilde{\mathbb{X}}(1)) = 0 | \Sigma = \sigma, \Theta_\Sigma = \theta_\sigma].$$

As earlier, $g_T(\theta_\sigma)$ can be evaluated by taking convolutions. Note that $g_T \neq g_M$ due to the restricted sublattice.

By conditioning on both Σ and Θ_Σ, one has

$$\begin{aligned} \bar{r} = \mathbb{P}[\varphi(X(1)) = 0] &= \mathbb{P}[\varphi(X(1)) = 0, \varphi(\Sigma) = 0] \\ &= \mathbb{E}[\mathbb{E}[I_{\{\varphi(\widetilde{\mathbb{X}}(1))=0\}} I_{\{\varphi(\Sigma)=0\}} | \Sigma, \Theta_\Sigma]] \\ &= \mathbb{E}[I_{\{\varphi(\Sigma)=0\}} g_T(\Theta_\Sigma)]. \end{aligned}$$

The hybrid simulation involves partitioning the edge set into F and $\bar{F}$. On the edge set F, let X^F be the random variable describing the state of all edges in F. Each simulation run consists of two steps:

1. *Cut*: An outcome of the random variable x^F is generated by independently cutting the edges $e \in F$ with probabilities q_e. This also gives a corresponding outcome σ of Σ. The complexity of this step is $O(|F|)$.
2. *Merge*: If the outcome σ is such that $\varphi(\sigma) = 1$, let $z = 0$. If the outcome σ is such that $\varphi(\sigma) = 0$, we generate a trajectory Θ_σ in $\mathbb{L}_{\sigma}^{\bar{F}}$, starting $(\widetilde{\mathbb{X}}(t))$ in state

σ. For an outcome θ_σ of Θ_σ, let $z = g_T(\theta_\sigma)$. The complexity of generating and calculating this is $O(|\sigma|^2)$.

Note that by the above construction, z is indeed an outcome of the random variable $Z = I_{\{\varphi(\Sigma)=0\}} g_T(\Theta_\Sigma)$. Hence, if $Z^{(1)}, \ldots, Z^{(N)}$ are independent samples from Z, then

$$\hat{\bar{r}} = \frac{1}{N} \sum_{i=1}^{N} Z^{(i)}$$

is an unbiased estimator of $\bar{r}$.

Figure 5 shows the sublattices of the complete four-node graph K_4. The lower left-hand corner shows the edge set partitions with F being a spanning tree (marked by thick lines). Depending on the outcome of the *Cut* step of the algorithm, the initial state of the Merge Process can be in any state in the sublattices marked by thick circles.

4.2. Tree Cut and Merge with Bounds

Assume that $\mathcal{G}$ is connected. If we choose to partition the edge set into a *Minimum Spanning Tree* T with respect to q_e, and its complementary set $\overline{T} = E \setminus T$, then T will connect the complete set of nodes V. Furthermore, if there are k failed links in T, they will partition the graph $\mathcal{G}(V, T)$ into exactly $k+1$ components. Let X be the random state of all edges, and let X^T be the random state of the edges in T. The network reliability can be expressed as

$$r = \sum_{k=0}^{n-1} \mathbb{P}[\varphi(X) = 1, |X^T| = k+1], \tag{11}$$

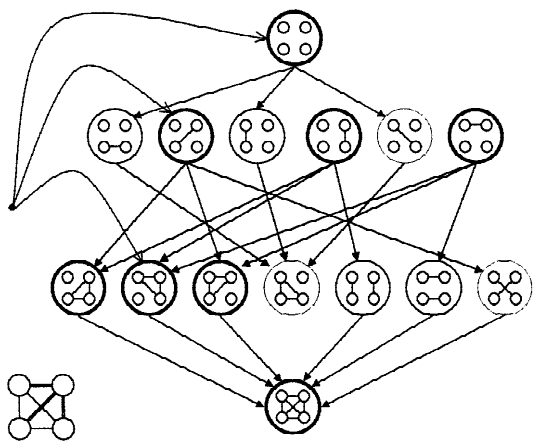

FIGURE 5. State transition diagram for Tree Cut and Merge scheme of K_4.

or

$$\bar{r} = \sum_{k=0}^{n-1} \mathbb{P}[\varphi(X) = 0, |X^T| = k+1], \tag{12}$$

where k is the number of cuts in tree T. Let $P_k = \mathbb{P}[|X^T| = k+1]$ be the probability of T having k cuts, and let $\overline{r_k} = \mathbb{P}[\varphi(X) = 0 | |X^T| = k+1]$ be the system's failure probability given there are k cuts in T. Then,

$$\bar{r} = \sum_{k=0}^{n-1} P_k \overline{r_k}.$$

Since $\mathcal{G}$ is connected and T connects $\mathcal{G}$, $\overline{r_0} = 0$ and, hence,

$$\bar{r} = \sum_{k=1}^{n-1} P_k \overline{r_k}$$

and

$$0 \le \bar{r} \le 1 - P_0,$$

where

$$P_0 = \prod_{i \in T} p_i.$$

Let $\overline{r_{k+}} = \sum_{i \ge k} \overline{r_i}$ and $P_{k+} = \sum_{i \ge k} P_i$. We can modify the Cut and Merge scheme to estimate $\overline{r_{1+}}$ as follows:

1. *Tree Cut 1+*: An outcome of the random state x^T given that there is at least one cut is generated by sequentially cutting the edges $e \in T$ as followed; see Appendix A.1 for details. For the ith edge in T:
   (a) If there are no failed edges before the ith edge, modify its failure probability to
   $$q_i' = \frac{q_i}{1 - \prod_{j \ge i} p_j}.$$
   (b) If there are any failed edges before the ith edge, keep its original failure probability q_i.

   This also gives a corresponding outcome σ of Σ. The complexity of this step is $O(n-1)$.
2. *Tree Merge*: Since the outcome σ is generated from cutting a spanning tree, it is certain that $\varphi(\sigma) = 0$. Next, we generate a trajectory Θ_σ in $\mathbb{L}_\sigma^{\overline{T}}$, starting $(\widetilde{\mathbb{X}}(t))$ in state σ. For an outcome θ_σ of Θ_σ, we calculate $g_T(\sigma, \theta_\sigma)$.

Let $z = g_T(\sigma, \theta_\sigma)$ be the outcome of each simulation run. Then, z is the outcome of the random variable $Z = \mathbb{P}[\varphi(X) = 0 | |X^T| \ge 1]$.

If we take N independent samples $Z^{(1)}, \ldots, Z^{(N)}$ from Z, then

$$\widehat{\overline{r_{1+}}} = \frac{1}{N} \sum_{i=1}^{N} Z^{(i)}$$

is an unbiased estimator of $\overline{r_{1+}}$, and $\bar{r}$ can be estimated by $\hat{\bar{r}} = P_{1+} \widehat{\overline{r_{1+}}}$.

4.2.1. Improving the bounds. There are only $n-1$ edges in T and, therefore, $n-1$ single cut states; as a consequence, P_1 can be calculated in $O(n)$ time. For each such state, there are only two components and, therefore, it takes at most one state jump to reach $\mathcal{U}$. $P_1 \overline{r_1}$ can easily be evaluated without convolution and the complexity is $O(n^2)$. Combining $\overline{r_1}$ with $\overline{r_0}$, an improved bound,

$$P_1 \overline{r_1} \le \bar{r} \le P_1 \overline{r_1} + P_{2+},$$

can be calculated in time $O(n^2)$. It is obvious that $P_{2+} = 1 - P_0 - P_1$, and we need only $\overline{r_{2+}}$ to complete the estimation of network failure probability. In modern communication systems, the p_e's are typically very high in fixed cable networks ($q_e < 10^{-6}$). It means that the above simple bounds will not be useless in most cases. More importantly, with a small amount of time invested, all of the remaining simulation effort can be channeled to estimating $\overline{r_{2+}}$ as follows:

1. *Tree Cut 2+*: An outcome of the random variable x^T given that there are at least two cuts is generated by sequentially cutting the edges $e \in T$ as follows (see Appendix A.2 for details). For the ith edge in T:
   (a) If there are no failed edges before the ith edge, modify its failure probability to

$$q_i' = \frac{q_i \left(1 - \prod_{j>i} p_j\right)}{1 - \left(1 + \sum_{j \ge i} \frac{q_j}{p_j}\right) \prod_{j \ge i} p_j}.$$

   (b) If exactly one edge failed before the ith edge, modify its failure probability to

$$q_i' = \frac{q_i}{1 - \prod_{j \ge i} p_j}.$$

   (c) If there are two or more edges failed before the ith edge, keep its original failure probability q_i.

   This also gives a corresponding outcome σ of Σ. The complexity of this step is $O(n-1)$.
2. *Tree Merge*: Since the outcome σ is generated from cutting a spanning tree, it is certain that $\varphi(\sigma) = 0$. Next, we generate a trajectory Θ_σ in $\mathbb{L}_\sigma^{\bar{T}}$, starting $(\tilde{\mathbb{X}}(t))$ in state σ. For an outcome θ_σ of Θ_σ, we calculate $g_T(\theta_\sigma)$.

Similar to that in the *Tree-Cut 1+* scheme, $z = g_T(\theta_\sigma)$ is an outcome of the random variable $Z = \mathbb{P}[\varphi(X) = 0 | |X^T| > 2]$. Hence, $\overline{r_{2+}}$ can be unbiasedly estimated by averaging N independent samples $Z^{(1)}, \ldots, Z^{(N)}$ from Z,

$$\widehat{\overline{r_{2+}}} = \frac{1}{N} \sum_{i=1}^{N} Z^{(i)},$$

and the estimate of $\bar{r}$ can be obtained from $\hat{\bar{r}} = P_1 \overline{r_1} + P_{2+} \widehat{\overline{r_{2+}}}$.

4.2.2. Performance of the Tree Cut and Merge (2+) scheme. The complexity of cutting the tree is $O(n)$, and the complexity of generating $g_T(\theta_\sigma)$ is $O(|\sigma|^2)$. Therefore, the mean complexity of estimating $\bar{r}$ is at most

$$C_1 n + C_2 \sum_{\sigma : |\sigma^T| > 2} \frac{\mathbb{P}[\Sigma = \sigma]}{P_{2+}} |\sigma|^2,$$

where C_1 and C_2 are constants.

5. NUMERICAL EXPERIMENTS

In this section, the Tree Cut and Merge algorithms are compared with the Standard Merge Process and the Leap–Evolve scheme. The *Relative Time Variance* product (RTV) is used as a metric to compare different algorithms, it is defined as the simulation time (in seconds) multiplied by the (estimated) squared relative error. For a large number of iterations N, the simulation time is proportional to N and the relative error is inversely proportional to $\sqrt{N}$. Therefore, the RTV is a number that largely depends on the network and the performance of the algorithm being studied rather than on N. The smaller the RTV value, the more efficient is the simulation algorithm.

An exact algorithm using the concept of connected components and exhaustive search is also implemented to confirm that the simulations produce accurate estimates. It evaluates (12) by exhausting all of the states in X within all possible cuts in T. A dodecahedron network (Fig. 6) with different link reliabilities is used to test the

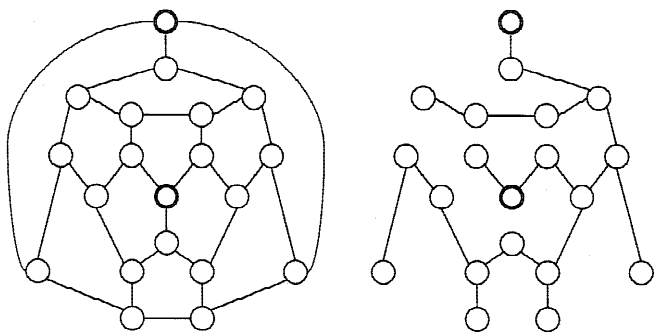

FIGURE 6. Dodecahedron network and its Minimum Spanning Tree.

TABLE 1. Descriptions of Different Labels Used

Label	Explanation
Tree Exact	Exact Evaluation algorithm
Standard Merge	Standard Merge Process algorithm
Leap–Evolve (0.15)	Leap–Evolve algorithm with a leap time of 0.15 s
Leap–Evolve (0.25)	Leap–Evolve algorithm with a leap time of 0.25 s
Tree–Merge (1+)	Tree Cut and Merge algorithm with one or more cuts in each sample
Tree-Merge (2+)	Tree Cut and Merge algorithm with two or more cuts in each sample
t	Total simulation time for 10^6 samples (in s)
Q_A	Estimated ALL-terminal network failure probability
Q_2	Estimated TWO-terminal network failure probability
re	Estimated relative error
RTV	Relative Time Variance product
bounds	Bounds on the network failure probability calculated by the algorithm

different schemes. In each experiment, all the Monte Carlo algorithms were run for 10^6 iterations and their results are listed for comparison. Table 1 lists the meaning of the labels used in the experimental results.

Experiment 1: ALL-terminal reliability of a heterogeneous unreliable network: In the first experiment, there are two groups of links: The first group is the minimum spanning tree (resembling a backbone network) and the second group consists of the remaining links with slightly lower reliability (resembling wireless backup links). In particular, the backbone links have failure probabilities of 0.1%, and the backup links have failure probabilities of 1%. The ALL-terminal network failure probability is to be estimated and the results are listed in Table 2. The best performing algorithm in this experiment is Tree-Merge (2+). In fact, the RTV of the

TABLE 2. ALL-Term Reliability of a Heterogeneous Unreliable Network

Scheme	t	Q_A	re	RTV	bounds
Tree Exact	1488	7.902e−7	n/a	n/a	n/a
Standard Merge	813	7.919e−7	1.48e−3	1.79e−3	n/a
Leap–Evolve (0.15)	108	7.885e−7	1.90e−3	3.89e−4	n/a
Leap–Evolve (0.25)	43	7.894e−7	2.94e−3	3.76e−4	n/a
Tree-Merge (1+)	53	7.854e−7	5.98e−3	1.89e−3	[0,1.883e−2]
Tree-Merge (2+)	82	7.908e−7	1.37e−3	1.55e−4	[6.915e−7,1.698e−4]

TABLE 3. TWO-Term Reliability of a Heterogeneous Unreliable Network

Scheme	t	Q_2	re	RTV	bounds
Tree Exact	3922	1.123e−7	n/a	n/a	n/a
Standard Merge	794	1.118e−7	4.73e−3	1.67e−2	n/a
Leap–Evolve (0.15)	70	1.123e−7	5.71e−3	2.29e−3	n/a
Leap–Evolve (0.25)	34	1.144e−7	8.29e−3	2.31e−3	n/a
Tree-Merge (1+)	25	1.140e−7	1.63e−2	6.55e−3	[0,1.883e−2]
Tree-Merge (2+)	46	1.124e−7	1.42e−3	9.30e−5	[1.002e−7,1.692e−4]

best Leap–Evolve algorithm (a leap time of 0.2 s) is 3.193×10^{-4} (not shown in the table), almost double that of the Tree-Merge (2+). It shows that the Tree-Merge (2+) algorithm is able to take advantage of the backbone network and compress the variance significantly.

Experiment 2: TWO-terminal reliability of a heterogeneous unreliable network: The second experiment uses the same network as in the first experiment. This time, we are estimating the TWO-terminal network failure probability (the two terminal nodes are marked by thick circles in Fig. 6). The results are listed in Table 3 and the best performing algorithm is again Tree-Merge (2+). It shows that this algorithm is performing just as well in the TWO-terminal case and it is reasonable to extend this assumption to K-terminal cases.

Experiment 3: ALL-terminal reliability of a heterogeneous reliable network: The third experiment is the same as the first experiment except the backbone network is much more reliable; the link failure probability is 10^{-6}, which is more realistic in shielded cable networks. The results are listed in Table 4 and the best performing algorithm is still Tree-Merge (2+). Note that the RTV value of this algorithm has orders of magnitude improvement over any other algorithm under investigation!

TABLE 4. ALL-Term Reliability of a Heterogeneous Reliable Network

Scheme	t	Q_A	re	RTV	bounds
Tree Exact	1488	7.041e−10	n/a	n/a	n/a
Standard Merge	812	7.031e−10	1.38e−3	1.55e−3	n/a
Leap–Evolve (0.15)	47	7.045e−10	2.43e−3	2.79e−4	n/a
Leap–Evolve (0.25)	32	7.049e−10	6.79e−3	1.47e−3	n/a
Tree-Merge (1+)	53	7.044e−10	1.30e−3	8.99e−5	[0,1.900e−05]
Tree-Merge (2+)	82	7.041e−10	6.08e−7	3.02e−11	[7.040e−10,8.750e−10]

TABLE 5. ALL-Term Reliability of a Homogeneous Unreliable Network

Scheme	t	Q_A	re	RTV	bounds
Tree Exact	1637	2.030e−5	n/a	n/a	n/a
Standard Merge	812	2.028e−5	9.65e−4	7.57e−4	n/a
Leap–Evolve (0.15)	177	2.032e−5	1.25e−4	2.77e−4	n/a
Leap–Evolve (0.25)	68	2.025e−5	1.69e−3	1.93e−4	n/a
Tree-Merge (1+)	55	2.078e−5	4.65e−2	1.20e−1	[0, 0.1738]
Tree-Merge (2+)	85	2.003e−5	1.36e−2	1.56e−2	[5.858e−6,1.528e−2]

Experiment 4: ALL-terminal reliability of a homogeneous unreliable network: In the last experiment, the network was homogeneous and unreliable. Each link had the same failure probability of 1%, and the ALL-terminal network failure probability was to be estimated. The results are listed in Table 5. In this experiment, the best performing algorithm is the Leap–Evolve scheme with a leap time of 0.25 s. It shows that in this network, the Leap–Evolve scheme successfully reduces the number of components to be merged after the "Leap" step without largely sacrificing the variance. However, only the Tree Cut and Merge scheme can provide reliability bounds that are available within a fraction of a second.

5.1. Summary of the Results

In heterogeneous networks, the Tree Cut and Merge scheme can indeed take advantage of the backbone to compress the sample variance and speed up the simulation at the same time. In homogeneous unreliable networks, the Tree Cut and Merge does not compress the sample variance as much as the optimal Leap–Evolve scheme. In this case, the Leap–Evolve scheme may provide a better speed up without sacrificing variance too much. However, one of the problems with the Leap–Evolve scheme is that of finding the optimal leap time. From Experiment 1 through Experiment 4, the best leap time has shifted from 0.25 s to 0.15 s and the leap time is the critical parameter in this scheme. For instance, if we choose a leap time of 0.5s in Experiment 3, the RTV for the Leap–Evolve scheme would be 0.439 (well above that of Standard Merge Process).

6. CONCLUSION AND FUTURE DIRECTIONS

In this article, we developed the Tree Cut and Merge hybrid scheme to improve the Standard Merge Process. It shows substantial performance improvement in heterogeneous networks, which are common in telecommunication networks. Unlike the Leap–Evolve hybrid scheme, the performance of the Tree Cut and Merge scheme is

not dependent on the a priori optimization of a key parameter. This makes the scheme much easier to apply. The benefits of the Tree Cut and Merge scheme are twofold. First, by exactly calculating P_0 and $P_1\overline{r_1}$, it provides reliability bounds in a very short time with very little overhead. Second, it compresses the sample variance by estimating the remaining conditional probability $\overline{r_{2+}}$, which will be scaled down by P_{2+} to form the final estimate. To make that possible, we use the technique of sequential sampling introduced in the Tree Cut step of the scheme; it allows us to unbiasedly sample the states of the tree given there are two or more links failed.

The bounding technique introduced in this article concludes at $\overline{r_1}$ in the exact calculation stage; however, it is possible to further compress the sample variance by evaluating up to $\overline{r_k}$ and estimating the $\overline{r_{(k+1)+}}$. The maximum number of states we need to search to calculate $\overline{r_k}$ is $\mathtt{Bin}(n-1,k)2^k$, the computational complexity of computing all the P_k is $O(n^2)$ and the complexity of the sequential sampling in the Tree Cut step is $O(n)$.

A close inspection of the Tree Cut and Merge scheme reveals that the problem of calculating the network failure probability, $\bar{r}$, is subdivided into n separate calculations by the formula $\bar{r} = \sum_{k=0}^{n-1} P_k \overline{r_k}$. This creates the opportunity to apply other techniques such as *Importance Sampling* to further reduce sample variance. In a forthcoming article, we will investigate the application of the Importance Sampling and other techniques to the Tree Cut and Merge scheme and show that the combination of these techniques overcomes the potential shortcomings of the Tree Cut and Merge scheme in homogeneous networks.

References

1. Barlow, R.E. & Proschan, F. (1975). *Statistical theory of reliability and life testing*. New York: Holt, Rinehart & Wilson.
2. Colbourn, C.J. (1987). *The combinatorics of network reliability*. New York: Oxford University Press.
3. Colbourn, C.J. & Harms, D.D. (1994). Evaluating performability: Most probable states and bounds. *Telecommunication Systems* 2: 275–300.
4. Easton, M.C. & Wong, C.K. (1980). Sequential destruction method for Monte Carlo evaluation of system reliability. *IEEE Transactions on Reliability* 29: 27–32.
5. Elperin, T., Gertsbakh, I., & Lomonosov, M. (1991). Estimation of network reliability using graph evolution models. *IEEE Transactions on Reliability* 40(5): 572–581.
6. Elperin, T., Gertsbakh, I., & Lomonosov, M. (1992). An evolution model for Monte Carlo estimation of equilibrium network renewal parameters. *Probability in the Engineering and Informational Sciences* 6: 457–469.
7. Fishman, G.S. (1986). A Monte Carlo sampling plan for estimating network reliability. *Operations Research* 34(4): 122–125.
8. Kumamoto, H., Tanaka, K., Inoue, K., & Henley, E.J. (1980). Dagger sampling Monte Carlo for system unavailability evaluation. *IEEE Transactions on Reliability* 29(2): 376–380.
9. Lomonosov, M. (1994). On Monte Carlo estimates in network reliability. *Probability in the Engineering and Informational Sciences* 8: 245–264.
10. Lomonosov, M. & Shpugin, Y. (1999). Combinatorics of reliability Monte Carlo. *Random Structures & Algorithms* 14: 329–343.
11. Provan, J.S. & Ball, M.O. (1982). The complexity of counting cuts and of computing the probability that a graph is connected. *SIAM Journal of Computing* 12: 777–787.

APPENDIX: UNBIASED SAMPLING OF A GRAPH GIVEN LINK FAILURES

Let $\mathcal{G}(V,E)$ be a graph with edge set $E = \{e_1,\ldots,e_m\}$, where edge e_i has a failure probability of q_i (or a functioning probability of $p_i = 1 - q_i$). If the edges are very reliable (i.e., $q_i \to 0$), the Crude Monte Carlo (CMC) sampling scheme is very inefficient in sampling states of such a network. The question is, can we modify the Crude Monte Carlo scheme to sample the network's rare states where at least one link has failed? Fishman used *Procedure Q* to sample edge states while avoiding certain cut-sets and path-sets [7]. It involves a sequential sampling technique which we can modify to fit our purpose.

Let Y_i be the binary random variable associated with edge e_i. In particular, $\{Y_i = 1\}$ is the event that the edge has failed, and $\{Y_i = 0\}$ is the event that the edge is operational.

A.1. Some Failed Links

We want to unbiasedly sample the network state given that there is at least one failed link (i.e., $\sum Y_i > 0$). It can be achieved through the sequential sampling technique, sampling the edge state one by one and modifying its failure probability according to the states of previous edges.

For the the ith edge in E, we have the following:

1. If there are no failed edges before the ith edge, the probability of e_i having failed given at least one edge has failed in E is

$$\mathbb{P}\left[Y_i = 1 \,\middle|\, \sum_{j\geq i} Y_j > 0\right] = \frac{\mathbb{P}\left[Y_i = 1, \sum_{j\geq i} Y_j > 0\right]}{\mathbb{P}\left[\sum_{j\geq i} Y_j > 0\right]}$$

$$= \frac{q_i}{1 - \prod_{j\geq i} p_j}.$$

2. If there is at least one failed edge before the ith edge, the probability of e_i having failed given at least one edge has failed in E is

$$\mathbb{P}\left[Y_i = 1 \,\middle|\, \sum_{j<i} Y_j > 0\right] = \frac{\mathbb{P}\left[Y_i = 1, \sum_{j<i} Y_j > 0\right]}{\mathbb{P}\left[\sum_{j<i} Y_j > 0\right]}$$

$$= q_i.$$

A.2. Two or More Failed Links

Having developed the sampling procedure for the situation in which at least one link has failed, we now develop the case for two or more failed links.

For the ith edge in E, we have the following:

1. If there are no failed edges before the ith edge, the probability of e_i having failed given at least two edges have failed in E is

$$\begin{aligned}
\mathbb{P}\left[Y_i = 1 \,\middle|\, \sum_{j\geq i} Y_j > 1\right] &= \frac{\mathbb{P}\left[Y_i = 1, \sum_{j\geq i} Y_j > 1\right]}{\mathbb{P}\left[\sum_{j\geq i} Y_j > 1\right]} \\
&= \frac{\mathbb{P}[Y_i = 1]\mathbb{P}\left[\sum_{j>i} Y_j > 0\right]}{\mathbb{P}\left[\sum_{j\geq i} Y_j > 1\right]} \\
&= \frac{q_i\left(1 - \prod_{j>i} p_j\right)}{1 - \mathbb{P}\left[\sum_{j\geq i} Y_i = 0\right] - \mathbb{P}\left[\sum_{j\geq i} Y_i = 1\right]} \\
&= \frac{q_i\left(1 - \prod_{j>i} p_j\right)}{1 - \prod_{j\geq i} p_j - \prod_{j\geq i} p_j \sum_{k\geq i} \frac{q_k}{p_k}} \\
&= \frac{q_i\left(1 - \prod_{j>i} p_j\right)}{1 - \left(1 + \sum_{k\geq i} \frac{q_k}{p_k}\right)\prod_{j\geq i} p_j}.
\end{aligned}$$

2. If there is exactly one failed edge before the ith edge, the probability of e_i having failed given at least two edges have failed in E is

$$\begin{aligned}
\mathbb{P}\left[Y_i = 1 \,\middle|\, \sum_{j\geq i} Y_j > 0\right] &= \frac{\mathbb{P}\left[Y_i = 1, \sum_{j\geq i} Y_j > 0\right]}{\mathbb{P}\left[\sum_{j\geq i} Y_j > 0\right]} \\
&= \frac{\mathbb{P}[Y_i = 1]}{1 - \mathbb{P}\left[\sum_{j\geq i} Y_j = 0\right]} \\
&= \frac{q_i}{1 - \prod_{j\geq i} p_j}.
\end{aligned}$$

3. If there are at least two failed edges before the ith edge, the probability of e_i having failed given at least two edges have failed in E is

$$\mathbb{P}\left[Y_i = 1 \,\middle|\, \sum_{j<i} Y_j > 1 \right] = q_i .$$

Indeed, the procedure can be extended to k or more failed links. However, we only use up to $k = 2$ in this article and, hence, have not shown the deductions for higher k.