Applications of the Cross-Entropy Method in Reliability

Dirk P. Kroese[♠] and Kin-Ping Hui[♥]

[♠] University of Queensland, Australia
[♥] Defence Science and Technology Organisation, Australia

1 Introduction

Telecommunication networks, oil platforms, chemical plants and airplanes consist of a great number of subsystems and components that are all subject to failure. Reliability theory studies the failure behavior of such systems in relation to the failure behavior of their components, which is often easier to analyze. However, even for the most basic reliability models, the overall reliability of the system can be difficult to compute. In this chapter we give an introduction to modern Monte Carlo methods for fast and accurate reliability estimation. We focus in particular on Monte Carlo techniques for network reliability estimation, and network design.

Network Reliability Estimation

It is well known that for large networks the exact calculation of the network reliability is difficult (indeed, this problem can be shown to be #P-complete [6,24]), and hence simulation becomes an option. However, in highly reliable networks such as modern communication networks, network failure is very infrequent, and direct simulation – also called *crude Monte Carlo* (CMC) simulation – of such rare events is computationally expensive. Various techniques have been developed to speed up the estimation procedure. For example, Kumamoto proposed a very simple technique called *Dagger Sampling* to improve the CMC simulation [20]. Fishman proposed *Procedure Q,* which can provide reliability estimates as well as bound [13]. Colbourn and Harms proposed a technique that will provide progressive bounds that will eventually converge to an exact reliability value [7]. Easton and Wong proposed a sequential construction method [10]. Elperin, Gertsbakh and Lomonosov proposed *Evolution*

Models for estimating the reliability of highly reliable networks [11, 12, 22]. Hui *et al*. [18] proposed a hybrid scheme that provides bounds and can provide a speed-up by several orders of magnitude in certain classes of networks. They also proposed another scheme [19] which employs the Cross-Entropy technique to speed-up the estimation in general classes of networks. Other relevant references on network reliability include [15, 31, 32]. We note that the network reliability in this chapter is always considered in the *static,* that is, non-repairable, case. However, for repairable systems a similar approach can be employed if instead of the system reliability the system *availability* is used. We will briefly discuss this issue in Section 2.

Network Design

Accurate reliability estimation is essential for the proper design, planning and improvement of an unreliable network, such as a telecommunications network. A typical question in network design is, for example, which components (links, nodes) to purchase, subject to a fixed budget, in order to achieve the most reliable network. There are several reasons why network planning is difficult. Firstly, the problem in question is a complex constrained integer programming problem. Secondly, for large networks the value of the objective function – that is, the network reliability – is difficult or impractical to evaluate [6, 24]. Thirdly, when Monte Carlo simulation is used to estimate the network reliability, the objective function becomes noisy (random). Finally, for highly reliable networks, sophisticated variance reduction techniques are required to estimate the reliability accurately. The literature on network planning is not extensive, and virtually all studies pertain to networks for which the system reliability can either be evaluated exactly, or sharp reliability bounds can be established. Colbourn and Harms [7] proposed a technique that provides progressive bounds that eventually converge to an exact reliability value. Cancela and Urquhart [4] employed a Simulated Annealing scheme to obtain a more reliable alternative network, given a user-defined network topology. Dengiz *et al*. used a Genetic Algorithm to optimize the design of communication network topologies subject to the minimum reliability requirement [9]. Yeh *et al*. [33] proposed a method based on a Genetic Algorithm to optimize the k-node set reliability subject to a specified capacity constraint. Reichelt *et al*. used a Genetic Algorithm in combination with a repair heuristic to minimize the network cost with specified network reliability constraints [25].

We show that the *Cross-Entropy* (CE) method [28] provides an effective way to solve both the network reliability estimation and the network planning/improvement problems. The CE method was first introduced in [26] as an adaptive algorithm for estimating probabilities of rare events in complex stochastic networks. It was soon realized [27, 29] that it could be used not only for rare event simulations but for solving difficult combinatorial optimization problems as well. Moreover, the CE method is well-suited for solving noisy optimization problems. A tutorial on the CE method can be found in [8], which is also available on-line from the CE homepage: `http://www.cemethod.org`.

The rest of the chapter is organized as follows. Network reliability is introduced in Section 2. Various Monte Carlo simulation techniques are described in Section 3. In Section 4, we present how the CE method can be applied to improve the Monte Carlo simulations. The more challenging problem of reliability optimization is tackled in Section 5 using the CE method. In Section 6, we show how to adapt the CE method to a generalized optimization problem in the context of network recovery and extension.

2 Reliability

The most basic mathematical model to describe complex unreliable systems is the following (see for example [1]): Consider a system that consists of m components. Each component is either functioning or failed. Suppose that the state of the system is also only functioning or failed. We wish to express the state of the system in term of the states of the components. This can be established by defining binary variables x_i, $i = 1,\dots,m$ representing the states of the components: $x_i = 1$ if the i-th component works, and 0 else. The state of the system, s say, is a binary variable as well (1 if the system works and 0 else). We assume that s is completely determined by the vector $\mathbf{x}=(x_1,\dots,x_m)$ of component states. In other words, we assume that there exists a function φ: $\{1,0\}^m\to\{1,0\}$ such that

$$s = \varphi(\mathbf{x}).$$

This function is called the *structure function* of the system. To determine the structure function of the system it is useful to have a graphical representation of system.

Example 1

Suppose a four-engine airplane is able to fly on just one engine on each wing. Number the engines 1, 2 (left wing) and 3, 4 (right wing). The network in Fig.1 represents the system symbolically. The structure function can be written as

$$\varphi(\mathbf{x}) = \big(1-(1-x_1)(1-x_2)\big)\big(1-(1-x_3)(1-x_4)\big).$$

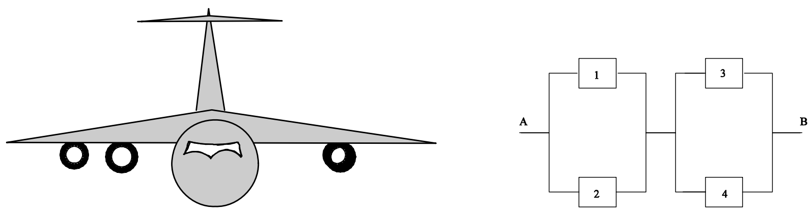

Fig. 1. An airplane with 4 engines, the system works if there is a "path" from A to B**.**

A system that only functions when *all* components are operational is called a *series* system. The structure function is given by

$$\varphi(\mathbf{x}) = \min\{x_1,\ldots,x_m\} = x_1 \cdots x_m = \prod_{i=1}^{m} x_i .$$

A system that functions as long as at least one component is operational is called a *parallel* system. Its structure function is

$$\varphi(\mathbf{x}) = \max\{x_1,\ldots,x_m\} = 1-(1-x_1)\cdots(1-x_m) \equiv \coprod_{i=1}^{m} x_i .$$

A *k*-out-of-*m* system is a system, which works if and only if at least *k* of the *m* components are functioning.

We mention two well-know techniques for establishing the structure function. The first is the *modular decomposition* technique. Often a system consists of combinations of series and parallel structures. The determination of the structure function for such systems can be handled in stages. The following example explains the procedure. Consider the upper-most system in Fig. 2. We can view the system as consisting of component 1 and modules 1* and 2*. This gives the second system in Fig. 2. Similarly, we can view this last system as a series system consisting of component 1 and module 1** (last system in Fig. 2).

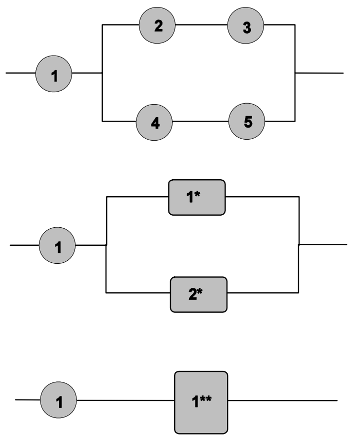

Fig. 2. Decomposition into modules

Now define s as the state of the system, z_1 as the state of module 1**, y_i the state of module i* and x_i the state of component i. Then, working "backwards", we have

$$s = x_1 z_1,$$
$$z_1 = 1-(1-y_1)(1-y_2),$$
$$y_1 = x_2x_3 \quad \text{and} \quad y_2 = x_4x_5.$$

Successive substitution gives

$$\varphi(\mathbf{x}) = s = x_1\left(1-(1-x_2x_3)(1-x_4x_5)\right).$$

A second technique for determining structure functions is the *method of paths and cuts*. Here, the structure function is assumed to be monotone, that is, $\mathbf{x} < \mathbf{y} \Rightarrow \varphi(\mathbf{x}) \le \varphi(\mathbf{y})$, for all vectors $\mathbf{x}$ and $\mathbf{y}$, where $\mathbf{x} < \mathbf{y}$ means that $x_i \le y_i$ for all $\boldsymbol{i}$ and $x_i < y_i$ for at least one $\boldsymbol{i}$. A *minimal path vector* (MPV) is a vector $\mathbf{x}$ such that $\varphi(\mathbf{x}) = 1$ and $\varphi(\mathbf{y}) = 0$ for all $\mathbf{y} < \mathbf{x}$. A *minimal cut vector* (MCV) is a vector $\mathbf{x}$ such that $\varphi(\mathbf{x}) = 0$ and $\varphi(\mathbf{y}) = 1$ for all $\mathbf{y} > \mathbf{x}$. The *minimal path set* corresponding to the MPV $\mathbf{x}$ is the set of indices i for which $x_i = 1$. The *minimal cut set* corresponding to the MCV $\mathbf{x}$ is the set of indices i for which $x_i = 0$.

The minimal path and cut sets determine the structure function. Namely, let $P_1,\ldots,P_p$ be the minimal path sets and $K_1,\ldots,K_k$ be the minimal cut sets of a system with structure function φ. Then,

$$\varphi(\mathbf{x}) = \coprod_{j=1}^{p} \prod_{i \in P_j} x_i,$$

and

$$\varphi(\mathbf{x}) = \prod_{j=1}^{k} \coprod_{i \in K_j} x_i .$$

The first equation is explained by observing that the system works if and only if there is at least one minimal path set with all components working. Similarly, the second equation means that the system working if and only if at least one component is working for each of the cut sets.

Example 2 (***Bridge Network***)

Consider the simple network in Fig.3, called a *bridge network*. The bridge network will serve as a convenient reference example throughout this chapter. Here we have five unreliable edges, labelled 1,…,5. The network is operating if the two terminal nodes A and B are connected by operational edges.

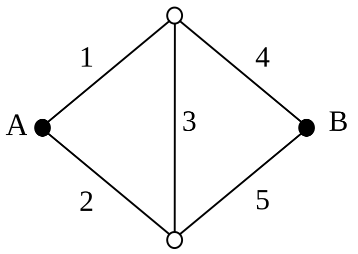

Fig. 3. Two-terminal bridge network

The minimal path sets are {1,4}, {2,5}, {1,3,5}, {2,3,4}, and the minimal cuts sets are {1,2}, {4,5}, {1,3,5}, {2,3,4}. The minimal path vector corresponding to the minimal path set {1,4} is the vector (1,0,0,1,0). The minimal cut vector corresponding to the minimal cut set {1,2} is the vector (0,0,1,1,1). It follows that the structure function $\boldsymbol{\varphi}$ is given by

$$\varphi(\mathbf{x}) = 1 - (1 - x_1 x_3 x_5)(1 - x_2 x_3 x_4)(1 - x_1 x_4)(1 - x_2 x_5). \tag{1}$$

2.1 Reliability function

We now turn to the case where the system's components are random. The *reliability* of a component is defined as the probability that the component will perform a required function under stated conditions for a stated period of time. Consider a system with *m* components and structure function φ, where the state of each component *i* is represented by a random variable X_i, with

$$X_i = \begin{cases} 1 & \text{with probability} \quad p_i \\ 0 & \text{with probability} \quad 1 - p_i \end{cases} \qquad i = 1, \ldots, m \, .$$

We gather the component reliabilities p_i into a vector $\mathbf{p} = (p_1,\ldots,p_m)$. The reliability of the system, i.e., the probability that the system works, is

$$\mathbb{P}[\text{System works}] = \mathbb{P}\left[\varphi(\mathbf{X}) = 1\right] = \mathbb{E}\left[\varphi(\mathbf{X})\right],$$

where $\mathbf{X}$ is the random vector $(X_1,\ldots,X_m)$. Under the assumption that the component states are independent, $\mathbb{P}[\varphi(\mathbf{X}) = 1]$ can be expressed in terms of $p_1,\ldots,p_m$. The function $r(\mathbf{p}) = \mathbb{P}[\varphi(\mathrm{X}) = 1]$ is called the *reliability function* of the system. Note that for the series and parallel systems the reliability function is $r(\mathbf{p}) = \prod_{i=1}^{m} p_i$ and $r(\mathbf{p}) = \coprod_{i=1}^{m} p_i$, respectively.

Example 3

For the bridge system we have by (1)

$$\varphi(\mathbf{X}) = 1 - (1 - X_1X_3X_5)(1 - X_2X_3X_4)(1 - X_2X_5)(1 - X_1X_4).$$

Using the fact that $X_\mathrm{i} = X_\mathrm{i}^2$, the expansion of $\varphi\,(\mathbf{X})$ can be written as

$$\begin{aligned}\varphi(\mathbf{X}) = {} & X_1X_3X_5 + X_2X_3X_4 + X_2X_5 + X_1X_4 - X_1X_2X_3X_5 \\ & - X_1X_2X_4X_5 - X_1X_3X_4X_5 - X_1X_2X_3X_4 - X_2X_3X_4X_5 \\ & + 2X_1X_2X_3X_4X_5.\end{aligned}$$

Because all the terms are products of independent random variables the reliability $r = r(\mathbf{p})$ is given by

$$\begin{aligned} r = {} & p_1p_3p_5 + p_2p_3p_4 + p_2p_5 + p_1p_4 - p_1p_2p_3p_5 - p_1p_2p_4p_5 \\ & - p_1p_3p_4p_5 - p_1p_2p_3p_4 - p_2p_3p_4p_5 + 2p_1p_2p_3p_4p_5.\end{aligned}$$

For highly reliable networks it is sometimes more useful to analyze or estimate the system *unreliability*

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

In this case the system unreliability is equal to

$$\bar{r} = q_1q_2 + q_2q_3q_4 - q_1q_2q_3q_4 + q_1q_3q_5 - q_1q_2q_3q_5 + q_4q_5$$
$$- q_1q_2q_4q_5 - q_1q_3q_4q_5 - q_2q_3q_4q_5 + 2q_1q_2q_3q_4q_5\,,$$

where $q_i = 1 - p_i$ is the unreliability of component i, $i = 1,\ldots,5$.

***Remark 1** (Availability)*

A concept closely related to reliability is the availability of a repairable system, defined as the long-run average fraction of the time that the system works.

Consider the simplest model of a repairable system where the system state is, as before, given by a structure function $\varphi(\mathbf{x})$. Suppose that each component i has a lifetime with cdf F_i and is being repaired according to a repair time cdf G_i. Assume that all the life and repair times are independent of each other. Note that the component state process $\{X_i(t), t \geq 0\}$ alternates between “up” and “down” (1 and 0), and forms a so-called *alternating renewal process.* The *availability,* $a_i(t)$, of component i time t is defined as the probability that it works at time t, that is, $a_i(t) = \mathbb{P}[X_i(t) = 1]$. From renewal theory the long-run average fraction of the time that i works, the *limiting availability* of i, is given by

$$a_i = \lim_{t\to\infty} a_i(t) = \lim_{t\to\infty} \mathbb{P}[X_i(t) = 1] = \frac{u_i}{u_i + d_i},$$

where u_i is the expected lifetime or *Mean Time To Failure* (MTTF), and d_i is the expected repair time or *Mean Time To Repair* (MTTR) of component i. Thus, the limiting availability depends only on the *means* of the distributions.

The system availability at time t, $a(t)$ say, is given by

$$a(t) = \mathbb{P}[\mathbf{X}(t) = 1] = r(a_1(t),\ldots,a_m(t)).$$

For each i, $a_i(t)$ converges to a constant a_i, as $t \to \infty$. Since r is a continuous function, we therefore have

$$a(t) \to r(a_1,\ldots,a_m),$$

as $t \to \infty$. This means that all the theory in this chapter for non-repairable or *static* systems can be applied to repairable systems, provided the notion of reliability is replaced by that of (limiting) availability.

2.2 Network Reliability

The reliability modeling of systems such as the airplane engines in Fig. 1 and the bridge network in Fig. 3 can be generalized to *network reliability systems* in the following way. Consider an undirected graph (or network) $\mathcal{G}(\mathcal{V},\mathcal{E},\mathcal{K})$, where $\mathcal{V}$ is the set of n vertices (or nodes), $\mathcal{E}$ is the set of m edges (or links), and $\mathcal{K} \subseteq \mathcal{V}$ is a set of *terminal* nodes, with $|\mathcal{K}| \geq 2$. Associated with each edge $e \in \mathcal{E}$ 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 $\mathbf{X} = (X_1,\ldots,X_m)$ the (failure) state of the network, or the state of the set $\mathcal{E}$. Let $\mathcal{S}$ be the set of all 2^m possible states of $\mathcal{E}$.

Notation A

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

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

We can identify $\mathbf{x}$ with the set $\mathcal{A}$. Henceforth we will use this identification whenever this is convenient.

Next, we assume that the random variables $\{X_e,\ e\in\mathcal{E}\}$ are mutually independent. Let p_e and q_e denote the reliability and unreliability of $e\in\mathcal{E}$ respectively. That is $p_e = \mathbb{P}[X_e = 1]$, and $q_e = \mathbb{P}[X_e = 0] = 1 - p_e$. Let $\mathbf{p} = (p_1,\ldots,p_m)$. The reliability $r = r(\mathbf{p})$ of the network is defined as the probability of $\mathcal{K}$ being connected by operational edges. Thus,

$$\begin{aligned} r &= \mathbb{E}\left[\varphi(\mathbf{X})\right] = \sum_{\mathbf{x}\in\mathcal{S}} \varphi(\mathbf{x})\,\mathbb{P}[\mathbf{X} = \mathbf{x}] \\ &= \sum_{\mathbf{x}\in\mathcal{S}} \varphi(\mathbf{x}) \prod_{i=1}^{m} \left[p_i^{x_i} (1-p_i)^{1-x_i} \right], \end{aligned} \tag{2}$$

where

$$\varphi(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathcal{K} \text{ is connected,} \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

3 Monte Carlo Simulation

The evaluation of network reliability in general is a #P-complete problem. When direct evaluation, e.g., via (2), is not feasible, estimation via Monte Carlo simulation becomes a viable option. The easiest way to estimate the reliability r (or unreliability $\bar{r}$) is to use CMC simulation, that is, let $\mathbf{X}_{(1)},\ldots,\mathbf{X}_{(N)}$ be independent identically distributed random vectors with the same distribution as $\mathbf{X}$. Then

$$\hat{r}_{\text{CMC}} = \frac{1}{N}\sum_{i=1}^{N}\varphi\left(\mathbf{X}_{(i)}\right)$$

is an unbiased estimator of r. Its sample variance is given by

$$\operatorname{var}\left(\hat{r}_{\text{CMC}}\right) = \frac{r\left(1-r\right)}{N}.$$

An important measure for the efficiency of an estimator $\hat{\ell}$ is its *relative error*, defined as $\operatorname{Std}\left(\hat{\ell}\right)\big/\mathbb{E}\left[\hat{\ell}\right]$. The relative error for $\hat{r}_{\text{CMC}}$ is thus given by

$$\operatorname{re}\left(\hat{r}_{\text{CMC}}\right) = \sqrt{\frac{\operatorname{var}\left(\hat{r}_{\text{CMC}}\right)}{\left(\mathbb{E}\left[\hat{r}_{\text{CMC}}\right]\right)^2}} = \sqrt{\frac{r\left(1-r\right)/N}{r^2}} = \sqrt{\frac{1-r}{Nr}}.$$

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

$$\operatorname{re}\left(\hat{\bar{r}}_{\text{CMC}}\right) = \sqrt{\frac{1-\bar{r}}{N\bar{r}}}.$$

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*. Next, we discuss methods to increase the accuracy of simulation procedures that work well for rare events.

3.1 Permutation Monte Carlo and the Construction Process

A more efficient way than CMC for estimating the static network unreliability is *Permutation Monte Carlo* simulation [11]. This approach can be applied to estimating equilibrium renewal parameters (see [21]) such as availability described in Remark 1. The idea is as follows. Consider the network $\mathcal{G}(\mathcal{V},\mathcal{E})$ in which each edge e has an exponential repair time with repair rate $\lambda(e) = -\ln(q_e)$ where q_e is the failure probability of e. That is, the repair time of edge e is exponentially distributed with mean $1/\lambda(e)$. We assume that at time t = 0 all edges are failed and 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 $\mathcal{E}$ at time t is given by the vector $\mathbf{X}(t)$, defined in a similar way as before. Thus, $(\mathbf{X}(t),\ t \geq 0)$ is a Markov process [2, 16, 23] with state space $\{0,1\}^m$ or, in view of Notation A, a Markov process on the subsets of $\mathcal{E}$. This process is called the *Construction Process* (CP) of the network.

Let Π denote the *order* in which the edges come up (become operational), and let S_0, $S_0+S_1,\ldots$, $S_0+\ldots+S_{m-1}$ be the times when those edges are constructed. Hence, the S_i are sojourn times of $(\mathbf{X}(t),\ t \geq 0)$. Π is a random variable which takes values in the space of permutations of $\mathcal{E}$ denoted by Ξ. For any permutation $\pi = (e_1,\ldots,e_m)$ define

$$\begin{aligned} \mathcal{E}_0 &= \mathcal{E}, \\ \mathcal{E}_i &= \mathcal{E}_{i-1} \setminus \{e_i\}, \quad 1 \leq i \leq m-1, \\ \lambda(\mathcal{E}_i) &= \sum_{e \in \mathcal{E}_i} \lambda(e), \end{aligned}$$

and let

$$\mathrm{b}(\pi) = \min_i \{\varphi(\mathcal{E} \setminus \mathcal{E}_i) = 1\}$$

be the *critical number* of π, that is, the number of repairs required to bring up the network. From the general theory of Markov processes it is not difficult to see that

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

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

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 equation (2). Hence, by conditioning on Π we have

$$r = \mathbb{E}\left[\varphi(\mathbf{X}(1))\right] = \sum_{\pi \in \Xi} \mathbb{P}[\Pi = \pi]\ \mathbb{P}\left[\varphi(\mathbf{X}(1)) = 1 \mid \Pi = \pi\right],$$

or

$$\bar{r} = 1 - r = \sum_{\pi \in \Xi} \mathbb{P}[\Pi = \pi]\ \mathbb{P}\left[\varphi(\mathbf{X}(1) = 0) \mid \Pi = \pi\right]. \tag{4}$$

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

$$\begin{aligned}\mathbb{P}\left[\varphi\left(\mathbf{X}(t)\right)=0 \mid \Pi=\pi\right] &= \mathbb{P}\left[S_0+\cdots S_{\mathrm{b}(\pi)-1}>t \mid \Pi=\pi\right] \\ &= 1-\underset{0\leq i<\mathrm{b}(\pi)}{\mathrm{Conv}}\left\{1-\exp\left[-\lambda\left(\mathcal{E}_i\right)t\right]\right\}.\end{aligned} \tag{5}$$

Let

$$\mathrm{g}_C(\pi)=\mathbb{P}\left[\varphi\left(\mathbf{X}(1)\right)=0 \mid \Pi=\pi\right], \tag{6}$$

as given in equation (5). Equation (4) can be rewritten as

$$\bar{r}=\mathbb{E}\left[\mathrm{g}_{\mathrm{C}}(\Pi)\right], \tag{7}$$

and this shows how the CP simulation scheme works. Namely, for $\Pi_{(1)},\ldots,\Pi_{(N)}$ independent identically distributed random permutations, each distributed according to Π, one has

$$\hat{\bar{r}}_{CP}=\frac{1}{N}\sum_{i=1}^{N} g_c\left(\Pi_{(i)}\right) \tag{8}$$

as an unbiased estimator for $\bar{r}$. We note that the CP estimation scheme is a particular instance of *conditional Monte Carlo* estimation. It is well known that conditioning always reduces variance; see for example Section 4.4 of [30]. As a result, the CP estimator has a smaller variance than the corresponding CMC estimator. However this accuracy comes at the expense of more complex computations.

3.2 Merge Process

A closer look at the evolution of the CP process reveals that many of the above results remain valid when we merge various states into “super states” at various stages of the process. This is known as the *Merge Process* (MP). We briefly describe the ideas below (see [22] for a detailed description and [21] for its application to estimating equilibrium renewal parameters)

To begin with, any subset $\mathcal{F} \subseteq \mathcal{E}$ of edges partitions the nodeset $\mathcal{V}$ into connected components known as a *proper partition*. Let σ = $\{\mathcal{V}_1,\ldots,\mathcal{V}_k\}$ be the proper partition of the subgraph $\mathcal{G}(\mathcal{V},\mathcal{F})$ (including isolated nodes, if any). Let I_i denote the edge-set of the induced subgraph $\mathcal{G}(\mathcal{V}_i)$. The set $I_\sigma = \mathrm{I}_1 \cup \ldots \cup \mathrm{I}_k$ of *inner* edges, that is, the edges within the components, is the *closure* of $\mathcal{F}$ (denoted by $\langle\mathcal{F}\rangle$). Denote its complement (the *inter-component* edges) by $\mathcal{E}_\sigma = \mathcal{E} \setminus I_\sigma$. Fig. 4 shows an example of a complete 6-node graph (K_6), a subgraph, and its corresponding closure.

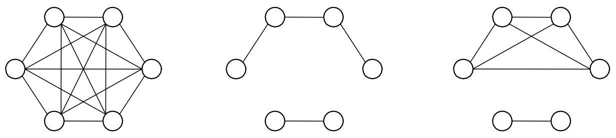
Fig. 4. K_6, a subgraph, and its corresponding closure.

Let $\mathbb{L}(\mathcal{G})$ be the collection of all proper partitions of $\mathcal{G}(\mathcal{V},\mathcal{E})$. The states in $\mathbb{L}(\mathcal{G})$ are ordered by the relation $\tau \prec \sigma \Leftrightarrow I_\tau \subset I_\sigma$ (that is, σ is obtained by merging components of τ). Any state σ in $\mathbb{L}(\mathcal{G})$ has a transition path to the terminal state $\sigma_\omega = \mathcal{G}$. Therefore $\mathbb{L}(\mathcal{G})$ is a *lattice*.

We consider now the CP $(\mathbf{X}(t))$ of the network. By restricting the process $(\mathbf{X}(\mathrm{t}))$ to $\mathbb{L}(\mathcal{G})$ we obtain another Markov process $(\mathbb{X}(\mathrm{t}))$, 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 $\mathcal{G}(\mathcal{V},\mathcal{E})$. Figure 5 shows $\mathbb{L}(K_4)$, the lattice of all proper partitions of the complete 4-node graph K_4, grouped into 4 different levels according to the number of components. The arrows show the direct successions in $\mathbb{L}(K_4)$, thus forming the transition graph of the Markov process $(\mathbb{X}(t))$.

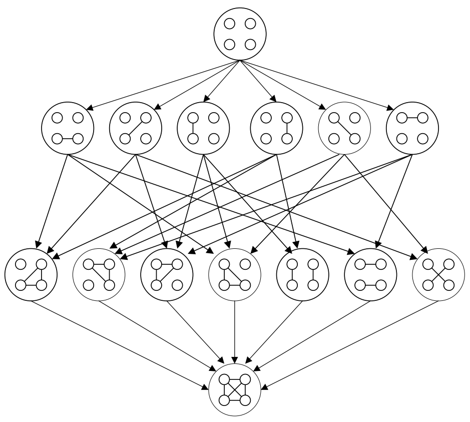

Fig. 5. State transition diagram for merge process of K_4.

For each $\sigma \in \mathbb{L}(\mathcal{G})$ the sojourn time in σ has an exponential distribution with parameter $\lambda(\sigma) = \sum_{e \in \mathcal{E}_\sigma} \lambda(e)$, 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 to the results for the CP, we define a *trajectory* of $(\mathbb{X}(t))$ as a sequence $\theta = (\sigma_0,\dots,\sigma_{\mathrm{b}})$, where $\mathrm{b} = \mathrm{b}(\theta)$ is the first index i such that σ_i is "up", that is, the network is operational. Since $\varphi(\mathbb{X}(t)) = \varphi(\mathbf{X}(t))$, we have

$$\bar{r} = \mathbb{P}\left[\varphi\left(\mathbb{X}(1)\right) = 0\right] = \mathbb{E}\left[\mathrm{g}_{\mathrm{M}}(\Theta)\right],$$

where Θ is the random trajectory of $(\mathbb{X}(t))$. For each outcome $\theta = (\sigma_0,\dots,\sigma_{\mathrm{b}})$ of Θ, $\mathrm{g}_{\mathrm{M}}(\theta)$ is given by

$$\begin{aligned} \mathrm{g}_{\mathrm{M}}(\theta) &= \mathbb{P}\left[\varphi\left(\mathbb{X}(1)\right) = 0 \mid \Theta = \theta\right] \\ &= \mathbb{P}\left[S_0 + \cdots + S_{\mathrm{b}(\theta)-1} \mid \Theta = \theta\right], \end{aligned} \tag{9}$$

where S_i is the sojourn time at σ_i. Therefore, $\mathrm{g}_{\mathrm{M}}(\theta)$ is given by the value

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

at $t = 1$. Let $\Theta_{(1)},\ldots,\Theta_{(N)}$ be independent identically distributed random trajectories distributed according to Θ. Then,

$$\hat{\bar{r}}_{\mathrm{MP}} = \frac{1}{N}\sum_{i=1}^{N} \mathrm{g}_{\mathrm{M}}\left(\Theta_{(i)}\right) \tag{10}$$

is an unbiased estimator for $\bar{r}$. It can be shown that the MP estimator has a smaller variance than the CP estimator, due to the state space reduction.

4 Reliability Estimation using the CE Method

Consider the bridge network in Example 2. We assume the typical situation where the edges are highly reliable, that is, the q_i are close to 0. The probability of the rare event $\{\varphi(\mathbf{X})=0\}$ is very small under CMC simulation and hence the accuracy of this sampling scheme is low. One way to combat the low accuracy problem is to “tilt” the probability mass function (pmf) of the component state vector $\mathbf{X}$ so that the rare event happens more often, and then multiply the structure function with a likelihood ratio to unbias the estimate. More precisely, let $\mathrm{f}(\mathbf{x}) = \mathbb{P}[\mathbf{X} = \mathbf{x}]$ be the original pmf of $\mathbf{X}$ and $\mathrm{g}(\mathbf{x})$ be a new pmf. Then,

$$\begin{aligned}\bar{r} &= \sum_{\mathbf{x}\in\mathcal{S}} \mathrm{f}(\mathbf{x})\left(1-\varphi(\mathbf{x})\right)\\ &= \sum_{\mathbf{x}\in\mathcal{S}} \mathrm{g}(\mathbf{x})\frac{\mathrm{f}(\mathbf{x})}{\mathrm{g}(\mathbf{x})}\left(1-\varphi(\mathbf{x})\right)\\ &= \mathbb{E}_{\mathrm{g}}\left[\frac{\mathrm{f}(\mathbf{X})}{\mathrm{g}(\mathbf{X})}\left(1-\varphi(\mathbf{X})\right)\right]\\ &= \mathbb{E}_{\mathrm{g}}\left[\mathrm{W}(\mathbf{X})\left(1-\varphi(\mathbf{X})\right)\right],\end{aligned}$$

where $\mathrm{W}(\mathbf{x})$ is the likelihood ratio for an outcome $\mathbf{x}$, and $\mathbb{E}_{\mathrm{g}}$ is the expectation under the pmf g. This indicates that we can estimate $\bar{r}$ also via

$$\frac{1}{N}\sum_{i=1}^{N}\mathrm{W}\left(\mathbf{X}_{(i)}\right)\left(1-\varphi\left(\mathbf{X}_{(i)}\right)\right),$$

where $\mathbf{X}_{(1)},\ldots,\mathbf{X}_{(N)}$ is a random sample from g. This is the well-know concept of *Importance Sampling* (IS).

Example 4 (Bridge Example (Continued))

Suppose that the edge failure probabilities are all the same, q_i=0.001. After choosing to “tilt” their probabilities to q_i’=0.5, the likelihood ratio becomes

$$\mathrm{W}(\mathbf{X}) = \frac{\prod_{i=1}^{5}\left(0.999X_i + 0.001(1-X_i)\right)}{0.5^5}.$$

Table 1 shows a simulation result with 10^5 samples. With a network failure probability of about two in a million, the CMC with 10^5 samples cannot estimate $\overline{r}$ accurately, while the CMC with IS (CMC-IS) delivers a much better result. This simple example demonstrates how Importance Sampling can help improve estimate accuracy.

Table 1. Results for CMC and CMC-IS

Scheme	$\widehat{\overline{r}}$	$\widehat{\mathrm{re}}$
CMC	0.000e-00	undefined
CMC-IS	2.022e-06	2.22e-00
True $\overline{r}$	2.002e-06	

However, the question of how we should tilt the parameters still remains open and that is where the CE technique can help. Before we discuss the CE technique, we first look at how to construct the ideal probability measure.

Consider the scenario where one wants to estimate the expectation ℓ of a positive function H(**X**). In the case of network reliability estimation, $\ell = \overline{r}$ and H(**X**) = 1–φ(**X**). It is possible to construct a probability measure such that one can accurately estimate ℓ with zero sample variance. Namely, since

$$\ell = \sum_{\mathbf{x}\in\mathcal{S}} \mathrm{f}(\mathbf{x})\mathrm{H}(\mathbf{x}) = \sum_{\mathbf{x}\in\mathcal{S}} \mathrm{f}(\mathbf{x})\mathrm{W}(\mathbf{x})\mathrm{H}(\mathbf{x}),$$

if we take $\mathrm{g}(\mathbf{x}) = \mathrm{g}^*(\mathbf{x}) \equiv \mathrm{f}(\mathbf{x})\mathrm{H}(\mathbf{x})/\ell$, then $\mathrm{W}(\mathbf{x}) = \ell/\mathrm{H}(\mathbf{x})$ and thus under g* we have $\mathrm{W}(\mathbf{X})\mathrm{H}(\mathbf{X}) = \ell$ with probability 1. In other words, we only need one sample from g* to obtain an exact estimate. Obviously such construction is of little practical use, as we need to know ℓ in order to construct g*(**x**). Moreover, f often comes from a parametric family f(**x**;**u**) where **u** is a parameter vector. One would like to keep g in the same family, that is g(**x**)=f(**x**;**v**) such that the likelihood ratio W(**x**;**u**,**v**)=f(**x**;**u**)/f(**x**;**v**) is easier to compute.

4.1 CE Method

It is not our intention to give a detailed account of the CE method for estimation – for this we refer to [8] and [28] – but in order to keep this chapter self-contained, we mention the main points. Consider the problem of estimating

$$\ell = \mathbb{E}_{\mathbf{u}}\left[\mathrm{H}(\mathbf{Y})\right] = \int \mathrm{H}(\mathbf{y})dF(\mathbf{y};\mathbf{u}). \tag{11}$$

Here H($\mathbf{y}$) is some positive function of $\mathbf{y} = (y_1,\ldots,y_m)$, and $F(\mathbf{y};\mathbf{u})$ is a probability distribution function with pmf or probability density function (pdf) f($\mathbf{y}$;$\mathbf{u}$) that depends on some *reference parameter* **u**. We consider for simplicity only the pdf case. The expectation operator under which the random vector $\mathbf{Y} = (Y_1,\ldots,Y_m)$ has pdf f($\mathbf{y}$;$\mathbf{u}$) is denoted by $\mathbb{E}_{\mathbf{u}}$. We can estimate ℓ using IS as

$$\hat{\ell} = \frac{1}{N}\sum_{i=1}^{N}\mathrm{H}\left(\mathbf{Y}_{(i)}\right)\mathrm{W}\left(\mathbf{Y}_{(i)};\mathbf{u},\mathbf{v}\right), \tag{12}$$

where $\mathbf{Y}_{(1)},\ldots,\mathbf{Y}_{(N)}$ is a random sample from f(·; **v**) – using a *different* reference parameter **v** – and

$$\mathrm{W}(\mathbf{Y};\mathbf{u},\mathbf{v}) = \frac{\mathrm{f}(\mathbf{Y};\mathbf{u})}{\mathrm{f}(\mathbf{Y};\mathbf{v})}, \tag{13}$$

Is the likelihood ratio. We can choose *any* reference vector **v** in equation (12) but we would like to use one that is in some sense “close” to the ideal (zero variance) IS pdf

$$\mathrm{g}^*(\mathbf{y}) = \frac{\mathrm{H}(\mathbf{y})\mathrm{f}(\mathbf{y};\mathbf{u})}{\ell}.$$

One could choose the parameter that minimizes the sample variance

$$\mathrm{var}_{\mathbf{v}}\left(\mathrm{H}(\mathbf{Y})\mathrm{W}(\mathbf{Y};\mathbf{u},\mathbf{v})\right)$$

through the optimization program

$$\min_{\mathbf{v}} \mathbb{E}\left[\mathrm{H}^2(\mathbf{Y})\mathrm{W}^2(\mathbf{Y};\mathbf{u},\mathbf{v})\right]. \tag{14}$$

The optimal solution of program (14) is typically hard to find and often not available analytically since the variance of H(**Y**)W(**Y**;**u**,**v**) is not either. One can view the variance as a “distance” measure, and the program (14) is to find a parameter vector **v** that minimizes the “variance distance” between g*(**y**) and g(**y**). An alternate distance measure is the *Kullback-Leiber*

CE distance (or simply CE distance). The CE distance between two probability densities f(**y**) and g(**y**) can be written as

$$\mathcal{D}(\mathrm{f},\mathrm{g}) = \int \mathrm{f}(\mathbf{y}) \ln \frac{\mathrm{f}(\mathbf{y})}{\mathrm{g}(\mathbf{y})} d\mathbf{y} = \int \ln \frac{\mathrm{f}(\mathbf{y})}{\mathrm{g}(\mathbf{y})} dF(\mathbf{y}).$$

The optimal reference parameter in the CE sense (that is, for which $\mathcal{D}$(g*,f(·;**v**)) is minimal) is then

$$\begin{aligned}\mathbf{v}^* &= \arg\min_{\mathbf{v}} \int \frac{\mathrm{H}(\mathbf{y})}{\ell} \ln \frac{\mathrm{H}(\mathbf{y})\mathrm{f}(\mathbf{y};\mathbf{u})}{\mathrm{f}(\mathbf{y};\mathbf{v})\ell} dF(\mathbf{y};\mathbf{u}) \\ &= \arg\min_{\mathbf{v}} \mathbb{E}_{\mathbf{u}}\left[\mathrm{H}(\mathbf{y}) \ln \frac{\mathrm{H}(\mathbf{y})\mathrm{f}(\mathbf{y};\mathbf{u})}{\mathrm{f}(\mathbf{y};\mathbf{v})}\right] \\ &= \arg\min_{\mathbf{v}} \left\{\mathbb{E}_{\mathbf{u}}\left[\mathrm{H}(\mathbf{y})\ln\left(\mathrm{H}(\mathbf{y})\mathrm{f}(\mathbf{y};\mathbf{u})\right)\right] - \mathbb{E}_{\mathbf{u}}\left[\mathrm{H}(\mathbf{y})\ln \mathrm{f}(\mathbf{y};\mathbf{v})\right]\right\}.\end{aligned}$$

Since H(**y**) ln(H(**y**) f(**y**;**u**)) does not depend on **v**, we have

$$\mathbf{v}^* = \arg\max_{\mathbf{v}} \mathbb{E}_{\mathbf{u}}\left[\mathrm{H}(\mathbf{y})\ln \mathrm{f}(\mathbf{y};\mathbf{v})\right]$$

Note that the expectation is under the original pdf f(**y**;**u**). However, we can apply the IS technique and use any pdf with parameter **w** to get the same optimal solution

$$\mathbf{v}^* = \arg\max_{\mathbf{v}} \mathbb{E}_{\mathbf{w}}\left[\mathrm{H}(\mathbf{y})\mathrm{W}(\mathbf{y};\mathbf{u},\mathbf{w})\ln \mathrm{f}(\mathbf{y};\mathbf{v})\right]. \tag{15}$$

Therefore, we can estimate the optimal CE reference vector as the solution of the iterative procedure

$$\mathbf{v}_t = \arg\max_{\mathbf{v}} \frac{1}{N}\sum_{i=1}^{N} \mathrm{H}\left(\mathbf{Y}_{(i)}\right)\mathrm{W}\left(\mathbf{Y}_{(i)};\mathbf{u},\mathbf{v}_{t-1}\right)\ln \mathrm{f}\left(\mathbf{Y}_{(i)};\mathbf{v}\right), \tag{16}$$

where at each iteration t a random sample from f(·;$\mathbf{v}_{t-1}$) is taken. The solution of program (16) can often be determined *analytically*. One example is when f is in an exponential family of distributions.

4.2 Tail Probability Estimation

In the *rare-event* setting, H(**Y**) is of the form H(**Y**) = I{S(**Y**)≥ γ} where I is the indicator function, γ is a constant, and

$$\ell = \mathbb{P}\left[\mathrm{S}(\mathbf{Y}) \geq \gamma\right] \tag{17}$$

is a small tail probability. The function S is called the *performance function*. For rare-event estimation problems, program (16) is difficult to carry out because the rareness of the event causes most of the indicators $H(\mathbf{Y}_{(i)})$ to be zero. For such problems, a two-phase CE procedure is employed: Apart from the reference parameter $\mathbf{v}$, we also choose to update the *level* γ, creating a sequence of pairs $\{(\mathbf{v}_t,\gamma_t)\}$ with the goal of estimating the optimal CE reference parameter $\mathbf{v}^*$. Starting with $\mathbf{v}_0 = \mathbf{u}$ (the original or nominal parameter vector), the updating formulas are as follows:

Given a random sample $\mathbf{Y}_{(1)},\ldots,\mathbf{Y}_{(N)}$ from $f(\cdot;\, \mathbf{v}_{t-1})$, we use the best performing ρ–portion of the samples. Let γ_t be the sample $(1-\rho)$–quantile of the performances $S(\mathbf{Y}_{(i)})$, $i = 1,\ldots,N$, provided the sample quantile is less than γ; otherwise we set γ_t equal to γ. In other words, set

$$\gamma_t = \min\left\{\gamma, S_{(\lceil(1-\rho)N\rceil)}\right\}, \tag{18}$$

where $S_{(j)}$ is the j-th *order-statistic* of the performances. Using the *same sample*, we let

$$\mathbf{v}_t = \arg\max_{\mathbf{v}} \frac{1}{N}\sum_{i=1}^{N} \mathrm{I}\left\{\mathrm{S}\left(\mathbf{Y}_{(i)}\right) \geq \gamma_t\right\} \mathrm{W}\left(\mathbf{Y}_{(i)};\mathbf{u},\mathbf{v}_{t-1}\right) \ln \mathrm{f}\left(\mathbf{Y}_{(i)};\mathbf{v}\right). \tag{19}$$

When γ_t reaches γ, we stop the iteration procedure and take $\mathbf{v}_t$ as the estimate of $\mathbf{v}^*$.

Again, it is important to understand that in many cases an explicit formula for $\mathbf{v}_t$ can be given, that is, we do not need to “solve” the optimization problem (19). Provided ρ is small and N is large enough, $\mathbf{v}_t$ in program (19) converges to the optimal $\mathbf{v}^*$ in program (15) (see [28]).

4.3 CMC and CE (CMC-CE)

The standard interpretation of the CMC scheme does not naturally allow for the application of the CE method. However, if we interpret CMC sampling using the CP framework, the CE method can be applied naturally. Instead of sampling the up/down state of individual edges, we can sample the up time of each edge. Then we check if the network is functioning at time t=1, and this probability is the network reliability estimate. In order to maintain the CMC features, we treat all edges independently and do not consider the concept of permutations.

In other words, we translate the original problem (estimating $\bar{r}$), which involves independent Bernoulli random variables $X_1,\ldots,X_m$, into an estimation problem involving independent exponential random variables

$Y_1,\dots,Y_m$. Specifically, imagine that we have a time-dependent system in which at time 0 all edges have failed and are under repair, and let $Y_1,\dots,Y_m$, with $Y_i \sim \mathsf{Exp}(u_i^{-1})$ and $u_i = 1/\lambda(i) = -1/\ln q_i$, be the independent *repair times* of the edges. Note that, by definition

$$\mathbb{P}\left[Y_i \geq 1\right] = \mathrm{e}^{-1/u_i} = q_i \quad i = 1,\dots,m.$$

Now, for each $\mathbf{Y} = (Y_1,\dots,Y_m)$, let $\mathrm{S}(\mathbf{Y})$ be the (random) time at which the system "comes up" (the terminal nodes become connected). Then, we can write

$$\bar{r} = \mathbb{P}\left[\mathrm{S}(\mathbf{Y}) \geq 1\right].$$

Hence, we have written the estimation of $\bar{r}$ in the standard rare event formulation of equation (17) and we can thus apply the CE method from [8], as described above. Note that $\gamma = 1$ in this situation.

Instead of sampling independently for each i from $\mathsf{Exp}(-1/u_i)$, we sample from $\mathsf{Exp}(-1/v_i)$. The vector $\mathbf{v} = (v_1,\dots,v_m)$ is thus our reference parameter. We now construct a sequence of pairs $\{(\mathbf{v}_t,\gamma_t)\}$ such that $\mathbf{v}_t$ converges to a reference vector close to the optimal CE reference parameter and γ_t eventually reaches one. Starting with $\mathbf{v}_0 = \mathbf{u} = (u_1,\dots,u_m)$, at each iteration t we draw a random sample $\mathbf{Y}_{(1)},\dots,\mathbf{Y}_{(N)}$ from the pdf $\mathrm{f}(\cdot;\mathbf{v}_{t-1}\})$ of $\mathbf{Y}$ and update the level parameter γ_t using equation (18) and the reference parameter $\mathbf{v}_t$ using equation (19), which in this case has the analytical solution

$$v_{t,j} = \frac{\sum_{i=1}^{N} I\left\{\mathrm{S}\left(\mathbf{Y}_{(i)}\right) \geq \gamma_t\right\} \mathrm{W}\left(\mathbf{Y}_{(i)};\mathbf{u},\mathbf{v}_{t-1}\right) \mathbf{Y}_{(i)j}}{\sum_{i=1}^{N} I\left\{\mathrm{S}\left(\mathbf{Y}_{(i)}\right) \geq \gamma_t\right\} \mathrm{W}\left(\mathbf{Y}_{(i)};\mathbf{u},\mathbf{v}_{t-1}\right)}, \tag{20}$$

where W is the likelihood ratio

$$\mathrm{W}\left(\mathbf{y};\mathbf{u},\mathbf{v}\right) = \frac{\mathrm{f}\left(\mathbf{y};\mathbf{u}\right)}{\mathrm{f}\left(\mathbf{y};\mathbf{v}\right)} = \exp\left(-\sum_{j=1}^{m} y_j \left(\frac{1}{u_j} - \frac{1}{v_j}\right)\right) \prod_{j=1}^{m} \frac{v_j}{u_j}.$$

After iteration T, when γ_T reaches one, we estimate $\bar{r}$ using IS as

$$\hat{\bar{r}} = \frac{1}{N} \sum_{i=1}^{N} I\left\{\mathrm{S}\left(\mathbf{Y}_{(i)}\right) \geq 1\right\} \mathrm{W}\left(\mathbf{Y}_{(i)};\mathbf{u},\mathbf{v}_T\right).$$

Example 5 (Bridge Network, CMC with CE)

Consider now the bridge network in Fig. 3. Suppose the "nominal" parameter vector is $\mathbf{u} = (0.3, 0.1, 0.8, 0.1, 0.2)$, that is $\mathbf{q}$ = (3.57e-2, 4.54e-5, 2.87e-1, 4.54e-5, 6.74e-3). A result of simulations is given in Table 2. The following CE parameters were used: (initial) sample size N=2000 and rarity parameter ρ=0.01 in equation (18). In both CMC and CMC-CE, a final sample size of one million was used to estimate $\bar{r}$.

Table 2. Results for CMC and CMC-CE

Scheme	$\hat{\bar{r}}$	$\widehat{\text{re}}$
CMC-CE	6.991e-05	1.67e-02
CMC	6.100e-05	1.28e-01
True $\bar{r}$	7.079e-05	

By using the CE method we have achieved, with minimal effort, a 98% reduction in variance compared to the CMC method. The CMC-CE algorithm required two iterations only to converge, as illustrated in Table 3. Notice that the algorithm tilted the parameters of the *mincut* elements {1,3,5} to higher values, which means they will fail much more often in the simulations. One can interpret this as the algorithm placing more importance on the mincut elements than on the remaining edges.

Table 3. Convergence of the parameters

t	γ_t	$\mathbf{v}_t$				
0	–	0.3	0.1	0.8	0.1	0.2
1	0.507	0.964833	0.216927	1.20908	0.0892952	0.567551
2	1.000	1.19792	0.120166	1.57409	0.0630103	1.15137

4.4 CP and CE (CP-CE)

We now apply the CE method to the CP simulation using reference parameters determined by the CE method rather than the nominal parameters. There are many ways to define a distribution on the space of permutations. However, note that the original distribution of Π is determined by the exponential distribution of $\mathbf{Y}$. In fact, Π can be viewed as a function of $\mathbf{Y}$. To see this, we generate $Y_1,\ldots,Y_m$ independently according to $Y_i \sim \mathsf{Exp}(u_i^{-1})$ and order the Y_i's such that $Y_{\Pi_1} \leq Y_{\Pi_2} \leq \cdots \leq Y_{\Pi_m}$. Then we take $\Pi(\mathbf{Y})$ = $(\Pi_1,\ldots,\Pi_m)$ as our random permutation.

We can express the network failure probability by

$$\bar{r} = \mathbb{E}_{\mathbf{u}}\left[g_C\left(\Pi\left(\mathbf{Y}\right)\right)\right] = \mathbb{E}_{\mathbf{u}}\left[\mathrm{S}\left(\mathbf{Y}\right)\right], \tag{21}$$

where we *redefine* S(**Y**) as $g_C(\Pi(\mathbf{Y}))$, with g_C being the Markov process function for the permutation defined in equation (6). A natural way of defining a change of measure is to choose different parameters v_i (instead of the nominal u_i) for the exponential distributions of the edge lifetimes, in a similar way to Section 4.3. Thus $\mathbf{v} = (v_1,\ldots,v_m)$ is still the vector of mean "repair" times. However, we have a slightly different situation from Section 4.3, because instead of having to estimate a rare event probability $\mathbb{P}[\mathrm{S}(\mathbf{Y}) \geq 1]$ we now have to estimate the (small) expectation $\mathbb{E}[\mathrm{S}(\mathbf{Y})]$. We can no longer use a *two-phase* procedure (updating γ and **v**), but instead use the one-phase procedure in which we only update $\mathbf{v}_t$. The analytic solution to program (16) for the *i*-th component of $\mathbf{v}_t$ is

$$v_{t,i} = \frac{\sum_{k=1}^{N} \mathrm{S}\left(\mathbf{Y}_{(k)}\right) \mathrm{W}\left(\mathbf{Y}_{(k)};\mathbf{u},\mathbf{v}_{t-1}\right) Y_{(k)i}}{\sum_{k=1}^{N} \mathrm{S}\left(\mathbf{Y}_{(k)}\right) \mathrm{W}\left(\mathbf{Y}_{(k)};\mathbf{u},\mathbf{v}_{t-1}\right)}, \tag{22}$$

where $Y_{(k)i}$ is the *i*-th component of $\mathbf{Y}_{(k)}$. To improve convergence in random sampling situations, it is often beneficial to use a smoothing parameter α to blend the old with the new estimates. That is we take

$$\mathbf{v}_t' = \alpha \mathbf{v}_t + \left(1-\alpha\right)\mathbf{v}_{t-1}$$

as the new parameter vector for the next iteration.

4.5 MP and CE (MP-CE)

The situation can be further generalized by employing CE for MP simulation. Recall that for each permutation π in the CP, there is a corresponding trajectory θ in the MP. Let Θ: $\pi \mapsto \theta$ be the mapping that assigns to each permutation $\boldsymbol{\pi}$ the corresponding unique trajectory θ. Then equation (21) can be rewritten as

$$\bar{r} = \mathbb{E}_{\mathbf{u}}\left[g_M\left(\Theta\left(\Pi\left(\mathbf{Y}\right)\right)\right)\right] = \mathbb{E}_{\mathbf{u}}\left[\mathrm{S}\left(\mathbf{Y}\right)\right],$$

where S(**Y**) has been redefined as $g_M(\Theta(\Pi(\mathbf{Y})))$, with g_M being the Markov process function for the trajectory θ defined in equation (9). The same CE procedure (22) described in Section 4.4 can be applied to the MP as well.

Example 6 (Bridge Network, MP and CP with CE)

We return to the bridge network of Example 5. Table 4 lists the results for the standard MP and CP simulations, compared with their counterparts with IS in which the reference parameters are determined by the CE method. The nominal reference parameter remains unchanged. That is, $\mathbf{u}$ = (0.3, 0.1, 0.8, 0.1, 0.2), and we use the CE parameters $\alpha = 0.7$ and N = 2000. The final sample sizes are $N = 10^5$ in all the original and CE simulations.

Table 4. Results for CP-CE and MP-CE

Scheme	$\hat{\bar{r}}$	$\widehat{\text{re}}$
MP-CE	7.082e-05	1.16e-03
MP	7.081e-05	1.32e-03
CP-CE	7.079e-05	1.21e-03
CP	7.079e-05	1.32e-03
CMC-CE	6.991e-05	1.67e-02
True $\bar{r}$	7.079e-05	

We have repeated this experiment numerous times and have consistently found that the Merge Process and the CP have very close performance in such a small example network. We also found that the CE technique provides an improvement (reduction) in variance of roughly 20% in both cases.

Note that the CMC simulation with CE still has over 100 times the variance of that in MP, MP-CE, CP or CP-CE simulations. This shows that no matter how much one modifies the CMC scheme with smart sampling techniques, the scheme still cannot compare to the simple CP sampling. In other words, it is the “structure” of sampling in the CP that makes it superior.

With the MP-CE or CP-CE sampling, there is no parameter γ to indicate when to stop the CE parameter tuning, therefore we need to use other strategies. Since we have imprecise knowledge of the performance function, we have to resort to simulation to evaluate that function at each point in order to optimize program (16). On the other hand, we do not want to spend too long on the CE parameter estimation effort, compared to the real simulation. As a result, we cannot use classic convergence criteria such as “stop when two consecutive vectors are ε close in some norm”. Fortunately, permutation (and trajectory) sampling depends on the relative weight of each edge and hence the sampling is fairly insensitive to the precise values of the Importance Sampling parameter $\mathbf{v}_t$. Therefore we only require a vector that is in the “right” region. As a rule of thumb, we recommend 5% to 10% of the final estimation effort to be spent on the CE parameter estimation.

Table 5 displays the evolution of the reference parameters for the MP-CE, where we stopped the CE algorithm after only three iterations, when the estimates “stabilized” (the values stop fluctuating). Again the algorithm allocated more attention to the mincut elements {1,3,5} and treated the rest as less important.

Table 5. Evolution of the reference parameters

t	$\mathbf{v}_t$				
0	0.3	0.1	0.8	0.1	0.2
1	0.35768	0.07378	0.86343	0.06899	0.2548
2	0.37752	0.06507	0.86312	0.05950	0.2718
3	0.38688	0.05956	0.85218	0.05764	0.2785

4.6 Numerical Experiments

In this section we give a few larger examples that might be found in communication networks. Fig. 6 shows a 3×3 and a 6×6 grid network, each network has four terminals at the corners. All links have the same failure probability. All experiments use a final sample size of 10^6 and the CE tuning batch sample size of 5000. In the tables, T denotes the CE tuning iteration and α denotes the smoothing parameter. The CMC-CE had a rarity parameter ρ=0.02. The estimated relative error ($\widehat{\mathrm{re}}$), sample variance ($\widehat{\mathrm{var}}$), simulation time (t) as well as Relative Time Variance (RTV) are also provided for comparisons. The RTV is defined as the product of the simulation time t (in seconds) and the estimated sample variance $\widehat{\mathrm{var}}$. It can be used as a metric to compare different algorithms. For a large number of samples N, the simulation time is proportional to N and the sample variance is inversely proportional to N. Therefore the RTV is a number that largely depends on the network and the performance of the algorithm under investigation rather than on N. The smaller the RTV value, the more efficient is the simulation algorithm.

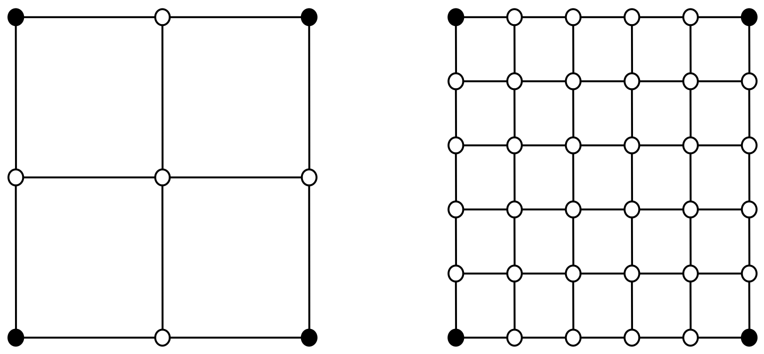

Fig. 6. A 3×3 and a 6×6 grid network

For verification purposes, the exact network failure probabilities are evaluated and listed as well. Note that in these trivial examples, alternative approaches such as approximation [3, 14] can also be used to obtain fairly accurate results.

Example 7 (3×3 unreliable grid)

In this example, all links of the 3×3 grid network have the same failure probability $q = 10^{-3}$. A result of the simulation is given in Table 6.

Table 6. Simulation results for the 3×3 unreliable grid network

Scheme	T	α	$\hat{\hat{r}}$	$\widehat{\text{re}}$	$\widehat{\text{var}}$	t	RTV
MP-CE	10	0.1	4.012e-06	1.07e-03	1.85e-17	18	3.34e-16
MP	-	-	4.012e-06	1.12e-03	2.03e-17	17	3.39e-16
CP-CE	10	0.1	4.011e-06	3.42e-03	1.88e-16	10	1.96e-15
CP	-	-	4.016e-06	3.89e-03	2.45e-16	9	2.24e-15
CMC-CE	4	1	2.830e-06	1.51e-01	1.81e-13	9	1.66e-12
CMC	-	-	4.000e-06	5.00e-01	4.00e-12	8	3.14e-11
True value			4.012e-06				

The CMC method gives a poor variance and relative error as expected. The CMC-CE shows a 95% reduction in variance but the variance is still too high to make this scheme very useful. In fact, the CMC-CE scheme has not converged in this example (also indicated by a relatively high $\widehat{\text{re}}$). The CP method gives a much smaller variance (0.1% of CMC-CE) while the CE method achieved a further reduction of 20-25% on average. The MP method has an even smaller variance (10% of CP) and the CE method provides roughly a 10% further reduction. Taking into account the computation overhead introduced by the CE method (approximately 10%), the MP-CE has a slight overall speed advantage over the MP algorithm, making the MP-CE the most efficient method to use.

Example 8 (6×6 reliable grid)

This is a larger network example consisting of 36 nodes and 60 edges with equal link failure probability q=10^{-6}. A result of the simulation is given in Table 7.

Table 7. Simulation results for the 6×6 reliable grid network

Scheme	T	α	$\hat{\bar{r}}$	$\widehat{\text{re}}$	$\widehat{\text{var}}$	t	RTV
MP-CE	10	0.1	3.999e-12	1.53e-03	3.76e-29	122	4.60e-27
MP	–	–	3.998e-12	1.75e-03	4.89e-29	113	5.55e-27
CP-CE	10	0.1	4.005e-12	1.28e-02	2.61e-27	66	1.72e-25
CP	–	–	4.006e-12	2.10e-02	7.07e-27	56	3.99e-25
CMC-CE	5	1	8.625e-14	9.08e-01	6.13e-27	41	2.51e-25
CMC	–	–	0	undefined	0	34	–
True value			4.000e-12				

The CMC and CMC-CE methods cannot handle such a low probability with a million samples. The CP provides good estimates and yet the CE method reduces the sample variances further by about 65% in the CP-CE. The MP starts with a much lower (1%) sample variance than that of CP and the CE further reduces it by 25% in the MP-CE. The RTV of the MP-CE and the CP-CE show a 20% and 130% speed up over the MP and the CP, respectively.

Example 9 (20-node 30-link unreliable network)

The next example is a 20-node 30-link network shown Fig. 7 with equal link failure probability of 3%. Two terminal reliability (the two terminal nodes are marked by thick circles in the figure) is to be estimated using different simulation schemes. A result of the simulation is given in Table 8. Note that in this example, the simple Minimum Cut Approximation method described in [6] will estimate a network failure probability of 5.4×10^{-5}, a 12% deviation from the true value of 6.138×10^{-5}.

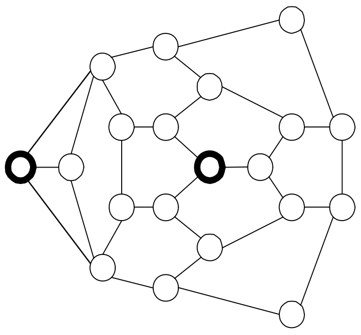

Fig. 7. A 20-node 30-link network

Table 8. Simulation results for the 20-node 30-link network

Scheme	T	α	$\hat{\bar{r}}$	$\widehat{re}$	$\widehat{var}$	t	RTV
MP-CE	10	0.1	6.128e-5	3.15e-3	3.72e-14	54	2.00e-12
MP	–	–	6.111e-5	3.99e-3	5.94e-14	50	2.94e-12
CP-CE	10	0.1	6.124e-5	1.13e-2	4.80e-13	26	1.24e-11
CP	–	–	6.192e-5	2.32e-2	2.06e-12	22	4.55e-11
CMC-CE	7	0.5	6.091e-5	6.46e-2	1.55e-11	18	2.80e-10
CMC	–	–	6.500e-5	1.24e-1	6.50e-11	16	1.04e-09
True value			6.138e-5				

The CMC method performed poorly as reflected in high variance and relative error. The CMC-CE method has significant improvement over CMC but is still far from ideal. Both CP and MP provide good estimates and the CE method improves them much further. In terms of RTV, the MP-CE and CP-CE have around 50% and 270% speed up over the MP and CP respectively.

4.7 Summary of results

With a better “sampling structure” and smart conditioning, the MP and CP schemes are superior to the CMC scheme. The CE technique further improves the performance of the MP and the CP schemes; the degree of improvement becomes more prominent as the network size grows. Close inspection of the IS parameter $\mathbf{v}_T$ reveals that the *bottleneck-cut* edges have been allocated higher importance than the rest.

Another point to note is the smoothing parameter α. If we keep $\alpha = 0.7$ as in the bridge example, the IS parameters $\mathbf{v}$ might oscillate instead of converge to the optimal $\mathbf{v}^*$ and as a consequence give poor estimates. We found that in larger networks, a smaller smoothing parameter such as $\alpha = 0.1$ is much more robust and always gave good results in our experiments. Numerical experience suggests that an increase in the tuning sample size N can alleviate the need to reduce the smoothing parameter α in larger problems. Of course, this means more effort has to be spent estimating each Importance Sampling parameter $\mathbf{v}_t$. However, if we leave α very small, more iterations are required for convergence towards $\mathbf{v}^*$. This raises the question of the most efficient way to allocate effort in estimating $\mathbf{v}^*$.

5 Network Design and Planning

This section is concerned with a network planning problem where the objective is to maximize the network's reliability subject to a fixed budget. More precisely, given a fixed amount of money and starting with a non-existent network, the question is which network links should be purchased, in order to maximize the reliability of the finished network. Each link carries a pre-specified price and reliability. This Network Planning Problem (NPP) is difficult to solve, not only because it is a constrained integer programming problem, which complexity grows exponentially in the number of links, but also because for large networks the value of the objective function – that is, the network reliability – becomes difficult or impractical to evaluate.

5.1 Problem Description

As before, consider a network represented as an undirected graph $\mathcal{G}(\mathcal{V},\mathcal{E})$, with set $\mathcal{V}$ of nodes (vertices), and set $\mathcal{E}$ of links (edges). The number of links is $|\mathcal{E}| = \mathrm{m}$. Without loss of generality we may label the links $1,\ldots,m$. Let $\mathcal{K} \subseteq \mathcal{V}$ be the set of terminal nodes. With each of the links is associated a *cost* c_e and reliability p_e. The objective is to buy those links that optimize the reliability of the network – defined as the probability that the terminal nodes are connected by functioning links – subject to a total budget $C_{\max}$. Let $\mathbf{c} = (c_1,\ldots,c_m)$ denote vector of link costs, and $\mathbf{p} = (p_1,\ldots,p_m)$ the vector of link reliabilities.

We introduce the following notation. For each link e let y_e be such that $y_e = 1$ if link e is purchased, and 0 otherwise. We call the vector $\mathbf{y} = (y_1,\ldots,y_m)$ the *purchase vector* and $\mathbf{y}^*$ the *optimal purchase vector*. Similarly, to identify the operational links, we define for each link ***e*** the link *state* by $x_e = 1$ if link e is bought and is functioning, and 0 otherwise. The vector $\mathbf{x} = (x_1,\ldots,x_m)$ is thus the state vector. For each purchase vector $\mathbf{y}$ let $\varphi_\mathbf{y}$ be the structure function of the purchased system. Thus, $\varphi_\mathbf{y}$ assigns to each state vector $\mathbf{x}$ the state of the system (working = terminal nodes are connected = 1, or failed = 0). Let X_e be random state of link e, and let $\mathbf{X}$ be the corresponding random state vector. Note that for each link e that is *not* bought, the state X_e is per definition equal to 0. The reliability of the network determined by $\mathbf{y}$ is (see Equation (2)) given by

$$r(\mathbf{y}) = \mathbb{E}\left[\varphi_\mathbf{y}(\mathbf{X})\right] = \sum_\mathbf{x} \varphi_\mathbf{y}(\mathbf{x})\, \mathbb{P}[\mathbf{X} = \mathbf{x}]. \tag{23}$$

We assume from now on that the links fail independently, that is, $\mathbf{X}$ is a vector of independent Bernoulli random variables, with success probability p_e for each purchased link e and 0 otherwise. Defining $\mathbf{p}_\mathbf{y} = (y_1 p_1,\ldots,y_m p_m)$, we write $\mathbf{X} \sim \mathsf{Ber}(\mathbf{p}_\mathbf{y})$. Our main purpose is to determine

$$\max_\mathbf{y} r(\mathbf{y}), \quad \text{subject to} \quad \sum_{e\in\mathcal{E}} y_e c_e \le C_{\max}\,. \tag{24}$$

Let $r^* \equiv r(\mathbf{y}^*)$ denote the optimal reliability of the network.

5.2 The CE Method for Combinatorial Optimization

The CE method is not only useful for rare-event estimation problems, but can also be applied to solve difficult discrete and continuous optimization problems. In this context, the method involves the following main steps, which are iterated:

1. Generate random states in the search space according to some specified random mechanism.
2. Update the parameters of this mechanism in order to obtain better scoring states in the next iteration. This last step involves minimizing the CE distance between two distributions.

We now specify these two steps for the NPP.

Random Network Generation

A simple method to generate the random purchase vectors is describe below: Let $\boldsymbol{a} = (a_1,\ldots,a_m)$ be the probability vector where a_e is the probability of purchasing edge e. Further let $\boldsymbol{Y}_{(k)}$ be the k-th random purchase vector where $Y_{(k),e}=1$ denotes edge e is purchased or else 0. Following is a simple algorithm to generate K random purchase vectors by rejecting the invalid (cost exceed maximum) ones.

Algorithm 1 (Generation Algorithm)

1. Generate a uniform random permutation $\pi = (e_1,\ldots,e_m)$. Set $k = 1$.
2. Calculate $C = c_{e_k} + \sum_{i=1}^{k-1} Y_{e_i} c_{e_i}$.
3. If $C \le C_{\max}$, draw $Y_{e_k} \sim \text{Ber}\left(a_{e_k}\right)$. Otherwise set $Y_{e_k} = 0$.
4. If $k = m$, then stop; otherwise set $k = k + 1$ and reiterate from step 2.

Remark 2

We note that, when drawing via Algorithm 1, the purchase vectors have some correlation bias. Theoretically, in order to generate random networks without such bias, one should sample from the conditional Bernoulli distribution (see [5]). However this is significantly more involved than the present algorithm/heuristic, and from our experience does not yield much gain.

Updating Generation Parameters

The usual CE procedure [28] proceeds by constructing a sequence of reference vectors $\{\mathbf{a}_t, t \ge 0\}$ (i.e., purchase probability vectors), such that $\{\mathbf{a}_t, t \ge 0\}$ converges to the degenerate (i.e., binary) probability vector $\mathbf{a}^*=\mathbf{y}^*$. The sequence of reference vectors is obtained via a two-step procedure, involving an auxiliary sequence of reliability levels $\{\gamma_t, t \ge 0\}$ that tend to the optimal reliability $\gamma^* = r^*$ at the same time as the $\mathbf{a}_t$ tend to $\mathbf{a}^*$. At each iteration t, for a given $\mathbf{a}_{t-1}$, γ_t is the sample $(1-\rho)$-quantile of performances (reliabilities). Typically ρ is chosen between 0.01 and 0.1. That is, generate a random sample $\mathbf{Y}_{(1)},\ldots,\mathbf{Y}_{(K)}$ using the generation algorithm above; compute the performances $r(\mathbf{Y}_{(i)})$, $I = 1,\ldots,K$ and let $\gamma_t = r_{(\lceil(1-\rho)K\rceil)}$, where $r_{(1)} \le \ldots \le r_{(K)}$ are the order statistics of the performances. The reference vector is updated via CE minimization, which (see [28]) reduces to the following: For a given fixed $\mathbf{a}_{t-1}$ and γ_t, let the j-th component of $\mathbf{a}_t$ be

$$a_{t,j} = \frac{\sum_{i=1}^{K} I\left\{r\left(\mathbf{Y}_{(i)}\right) \geq \gamma_t\right\} Y_{(i)j}}{\sum_{i=1}^{K} I\left\{r\left(\mathbf{Y}_{(i)}\right) \geq \gamma_t\right\}}, \quad j = 1, \ldots, m, \tag{25}$$

where we use the *same* random sample $\mathbf{Y}_{(1)},\ldots,\mathbf{Y}_{(k)}$ and where $Y_{(i)j}$ is the j-th coordinate of $\mathbf{Y}_{(i)}$.

The main CE algorithm for optimizing Equation (24) using the above generation algorithm is thus summarized as follows.

Algorithm 2 (Main CE Algorithm for Optimization)

1. Initialize a_0. Set t = 1 (iteration counter).

2. Generate a random sample $\mathbf{Y}_{(1)},\ldots,\mathbf{Y}_{(K)}$ using Algorithm **Error! Reference source not found.**, with $\mathbf{a} = \mathbf{a}_{t-1}$. Compute the sample (1–$\rho$)-sample of performances γ_t.

3. Use the *same* sample to update $\mathbf{a}_t$, using Equation (25).

4. If $\max\left(\min\left(\mathbf{a}_t, 1-\mathbf{a}_t\right)\right) \leq \beta$ for some small fixed β then stop (let T be the final iteration); otherwise set $t = t + 1$ and reiterate from step 2.

Noisy Optimization

As mentioned earlier, for networks involving a large number of links the exact evaluation of the network reliability is in general not feasible, and simulation becomes a viable option.

In order to adapt Algorithm 2 to *noisy* NPPs, we again, at iteration t, generate a random sample $\mathbf{Y}_{(1)},\ldots,\mathbf{Y}_{(N)}$ according the $\mathsf{Ber}\left(\mathbf{a}_{t-1}\right)$ distribution. However, the corresponding performances (network reliabilities) are now not computed exactly, but estimated by means of Monte Carlo simulations such as Equation (10). During the optimization process, one might need to estimate the reliability of a large number of networks using a limited number of samples. An efficient estimation algorithm is the MP described in previous section. It works well with relatively small sample size even for highly reliable networks.

5.3 Numerical Experiment

To illustrate the effectiveness of the proposed CE approach, consider the 6-node fully-connected graph with 3 terminal nodes given in Figure 8. The links costs and reliabilities are given in Table 9. Note that the direct links between the terminal nodes have infinite costs. We have deliberately excluded such links to make the problem more difficult to solve. The total budget is set to $C_{\max} = 3000$.

Note that for a typical purchase vector **y** the network reliability $r(\mathbf{y})$ will be high, since all links are quite reliable. Consequently, to obtain an accurate estimate of the network reliability, or better, the network unreliability $\overline{r}(\mathbf{y}) = 1 - r(\mathbf{y})$, via conventional Monte Carlo methods, would require a large simulation effort. The optimal purchase vector for this problem – computed by brute force – is $\mathbf{y}^*$ = (1,1,0,0,1,0,1,1,0,1,0,0,0,0,1), which yields a minimum network unreliability of $\overline{r}^* = 7.9762 \times 10^{-5}$.

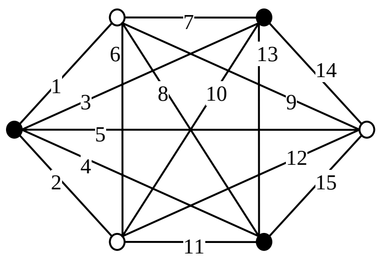

Fig. 8. Network with 3 terminal nodes, denoted by black vertices.

Table 9. Link costs and reliabilities

i	c_i	p_i	i	c_i	p_i	i	c_i	p_i
1	382	0.990	6	380	0.998	11	397	0.990
2	392	0.991	7	390	0.997	12	380	0.991
3	∞	0.992	8	395	0.996	13	∞	0.993
4	∞	0.993	9	396	0.995	14	399	0.992
5	320	0.994	10	381	0.999	15	392	0.994

We used the following parameters for our algorithm: the sample size in Step 2 of the CE algorithm $K = 300$; the sample size in Equation (10) $N = 100$; the initial purchase probability $\mathbf{a}_0$ = (0.5,…,0.5). The algorithm stops when all elements of $\mathbf{a}_t$ are less than $\beta = 0.01$ away from either 0 or 1. Let T denote the final iteration counter. We round $\hat{\mathbf{a}}_T$ to the nearest binary vector and take this as our solution **a*** to the problem.

Table 10 displays a typical evolution of the CE method. Here, *t* denotes the iteration counter, γ_t the sample (1–ρ)-quantile of the estimated unreliabilities, and $\mathbf{a}_t$ the purchase probability vector, at iteration *t*. The important thing to notice is that $\mathbf{a}_t$ quickly converges to the optimal degenerate vector $\mathbf{a}^* = \mathbf{y}^*$. The simulation time was 154 seconds on a 3.0GHz computer using a Matlab implementation.

In repeated experiments, the proposed CE algorithm performed effectively and reliably in solving the noisy NPP, which constantly obtained the optimal purchase vector. Moreover, the algorithm only required on average 9 iterations with a CPU time of 180 seconds.

Table 10. A typical evolution of the purchase vector

t	γ_t	$\mathbf{a}_t$														
0		0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50
1	4.0e-03	0.66	0.69	0.15	0.15	0.62	0.48	0.59	0.64	0.38	0.62	0.52	0.38	0.15	0.41	0.62
2	2.6e-04	0.69	0.63	0.05	0.05	0.72	0.21	0.88	0.71	0.33	0.75	0.58	0.26	0.05	0.38	0.77
3	1.4e-04	0.67	0.75	0.01	0.01	0.78	0.11	0.89	0.89	0.12	0.76	0.57	0.22	0.01	0.44	0.77
4	1.0e-04	0.76	0.76	0.00	0.00	0.89	0.03	0.97	0.90	0.06	0.83	0.43	0.11	0.00	0.41	0.84
5	8.1e-05	0.79	0.88	0.00	0.00	0.97	0.01	0.99	0.97	0.02	0.90	0.15	0.03	0.00	0.33	0.95
6	6.7e-05	0.94	0.96	0.00	0.00	0.97	0.00	1.00	0.99	0.01	0.97	0.07	0.01	0.00	0.10	0.99
7	6.3e-05	0.98	0.99	0.00	0.00	0.99	0.00	1.00	1.00	0.00	0.99	0.02	0.00	0.00	0.03	1.00
8	5.8e-05	0.99	1.00	0.00	0.00	1.00	0.00	1.00	1.00	0.00	1.00	0.01	0.00	0.00	0.01	1.00

6 Network Recovery and Expansion

This section looks at the optimal network design problem in an incremental sense, that is, one starts with a *baseline* network and has to decide which additional links should be bought, in order to optimally improve the reliability of the network. This situation occurs for example in military networks, where as a result of components failures or attacks, part of the network has become isolated and must be reconnected. Another example is when new nodes are being deployed, and one has to choose between many available options to connect them to the existing network.

6.1 Problem Description

Consider an existing network (may be non-functional) as a base network. Additional links are bought to improve the network reliability. In situations where multiple possible configurations are available, the goal is to find the configuration those results in highest network reliability. Let $\mathcal{E}_B$ denotes the edge set of the base network and $\mathcal{E}_i$ denotes the additional links in the i-th network $\mathcal{G}_I = \mathcal{G}(\mathcal{V},\mathcal{E}_B \cup \mathcal{E}_i,\mathcal{K})$. Here we assume the node set and terminals are the same among all the networks. If r_i denotes the reliability of the network $\mathcal{G}_i$, then the goal is to find the network $\mathcal{G}_o$ among all possible configurations such that

$$r_o = \max_i \{r_i\}.$$

In some situations where the connection points are limited – for example satellite terminals may be available to only a few nodes – the choices of alternate bearer (link) locations may not be extensive. In that case, the simplest approach is to use exhaustive search. Thus, the process can be divided into two steps:

1. generate all the valid configurations (or candidate networks $\mathcal{G}_i$); and
2. compare their reliabilities and find the optimal network $\mathcal{G}_o$.

Figure 9 shows an example network being separated into two groups. In order to rejoin the two groups, at least one link is needed and there are 12 possible ways to connect the node sets {A1, A2, A3} and {B1, B2, B3, B4}. It is easy enough to generate all the 12 candidate networks and compare their reliability to find the optimal location of the alternate bearer.

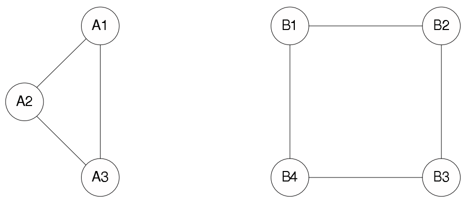

Fig. 9. Example with small number of candidates

If the existing network is large and/or multiple links are being added, the number of possible candidate networks grows very quickly. Often the computational cost of enumerating all networks and comparing their reliability becomes prohibitive. Another approach is to use simulation-based combinatorial optimization techniques under multiple constraints. The constraints of the optimization program can be as simple as fixing the maximum number of additional links, or specifying some maximum limit on the total cost of adding the links.

6.2 Reliability Ranking

As for large networks the exact calculation of network reliability is difficult, estimating it via Monte Carlo simulation becomes favourable. Note that in applications like network reliability optimization, one needs to compare the reliability of multiple *similar* networks. When simulation is used to estimate network reliability, sampling error (noise) is introduced. If the two networks are similar, the noise can become so significant that it may impair the accuracy of the comparisons. Hui *et al*. proposed a coupled approach to estimate reliability difference of two similar networks with high confidence [17]. In this section we demonstrate how a similar concept can be used and specialized to compare many similar networks very effectively. The concept is somewhat similar to using common random numbers to reduce variance in Monte Carlo simulation.

Edge Relocated Networks

The concept of *Edge Relocated Networks* [17] refers to networks having the same number of edges with matched link reliabilities[1]. The differences between the networks are thus restricted to a few links being reconnected to different nodes. In other words, if $\mathcal{G}_i$ and $\mathcal{G}_j$ are edge relocated networks, they share the same edge reliability vector $\mathbf{p}$. Figure 10 shows two edge relocated networks derived from the same base network.

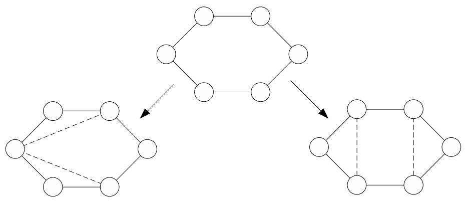
Fig. 10. Example of Edge Relocated Networks

Coupled Sampling

Since the edge relocated networks share the same edge reliability vector $\mathbf{p}$, it is proven [17] that their reliability can be compared very efficiently. The idea is to sample the edge states and observe them in different edge relocated networks. When dealing with many edge-relocated networks simul-

[1] Note that if there are unmatched links between the networks, redundant self-loops can be introduced to bring the networks to edge relocated versions of each other (see [17]).

taneously, one sampling scheme is of particular interest, namely *edge permutation sampling*.

Edge permutation sampling starts from the Construction Process in Section 3.1. In particular, we imagine the network is constructed dynamically by repairing links independently and with an exponential repair time. At time $t = 0$ each edge e is failed and is being repaired at a repair rate $\lambda(e) = -\ln(q_e)$. The network unreliability is equal to the probability of the dynamic network not being operational at time $t = 1$, and is of the form (see Equation (7)) $\overline{r} = \mathbb{E}\left[\mathrm{g}(\Pi)\right]$, where g is a know function involving convolutions and Π is a permutation describing the random order in which the links come up. By sampling from Π, $\overline{r}$ can be efficiently estimated via Equation (8).

This edge permutation sampling (or simply permutation sampling) scheme is superior to other combinatorial sampling schemes because it elegantly avoids the rare event problem. In highly reliable networks such as communication networks, the networks are functioning most of the time, and hence it is hard for the combinatorial schemes to sample the failure state and estimate its probability. As a consequence, it is more difficult to compare the reliability of similar networks. In permutation sampling, however, the networks always start at the same failure state and will eventually come up. The only question is when will they come up or what is their operational probability at time t=1.

When comparing reliability of networks, the *Coupled CP* proposed in [17] is very efficient in finding the reliability difference of two networks. It can achieve 10^{11} times speedup over the best known independent sampling scheme. The scheme uses the simple observation of the following equation

$$\overline{r_{i,j}} = \overline{r}_i - \overline{r}_j = \sum_{\pi} \mathbb{P}\left[\Pi = \pi\right]\left(g_i(\pi) - g_j(\pi)\right) = \mathbb{E}\left[g_i(\Pi) - g_j(\Pi)\right].$$

The method takes samples of permutations and observes how the two networks behave under the same edge permutation. Figure 11 shows an example edge construction sequence on two edge-relocated networks. Assuming the black nodes are the terminals, network $\mathcal{G}_A$ will come up (i.e. all terminal nodes become connected) on the 5-th edge while network $\mathcal{G}_B$ will be on the 6-th edge. Hence their probability functions $\mathrm{g}_A(\pi)$ and $\mathrm{g}_B(\pi)$ will be different.

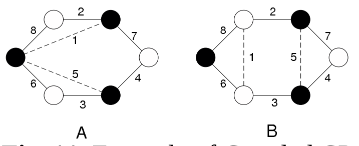

Fig. 11. Example of Coupled CP

If we take N random permutations $\{\Pi_{(1)},\ldots,\Pi_{(N)}\}$ and observe them on both networks, their reliability difference can be estimated by

$$\widehat{r_{A,B}} = \sum_{i=1}^{N} \frac{g_A\left(\Pi_{(i)}\right) - g_B\left(\Pi_{(i)}\right)}{N}.$$

Synchronous Construction Ranking (*SCR*)

In our reliability optimization problem, the prime interest is searching for the most reliable network among the candidates. Therefore the actual difference in reliability is not our main concern. All we need to find is which network is more reliable than others.

In the edge permutation sampling scheme, the most reliable network is expected to come up earlier than others. Therefore one can sample edge permutations and observe how all the candidate networks evolve simultaneously. The most reliable network should come up first most often.

Let $\mathrm{b}_i(\pi)$ denotes the critical number of graph $\mathcal{G}_i$ on permutation π, that is the ordinal number when the network comes up. For example in Figure 11, $\mathrm{b}_A(\pi) = 5$ and $\mathrm{b}_B(\pi) = 6$. Let $\mathrm{b}^*(\pi)$ be the smallest critical number among the candidate networks on a given permutation π, that is

$$\mathrm{b}^*(\pi) = \min_i \mathrm{b}_i(\pi).$$

Then $\mathcal{G}_o$ is the most reliable network if and only if it has the highest chance of coming up before any other candidates. Mathematically, it is the network that corresponds to the solution of the program

$$\max_i \mathbb{P}\left[\mathrm{b}_i(\Pi) = \mathrm{b}^*(\Pi)\right].$$

This is how the SCR scheme works: First randomly sample N permutations and then estimate the probability of network $\mathcal{G}_i$ being the best by

$$P_o\left(\mathcal{G}_i\right) = \sum_{j=1}^{N} \frac{I\left\{\mathrm{b}_i\left(\Pi_{(j)}\right) = \mathrm{b}^*\left(\Pi_{(j)}\right)\right\}}{N}.$$

Finally, find the network with the maximal P_o and take it as the optimal $\mathcal{G}_o$.

Example 10

Figure 12 shows an example of a surviving network that consists of 18 nodes and 28 links. {C1,…,C5} are the core routers and {A0,…,A12} are the access routers. A0 is currently isolated and needs to be re-connected to the network. Assuming there are enough resources to provide two wireless links connecting any nodes, the question is where the extra links should be attached in order to achieve maximal all-terminal reliability. Table 11 lists the reliabilities of different types of link in the example.

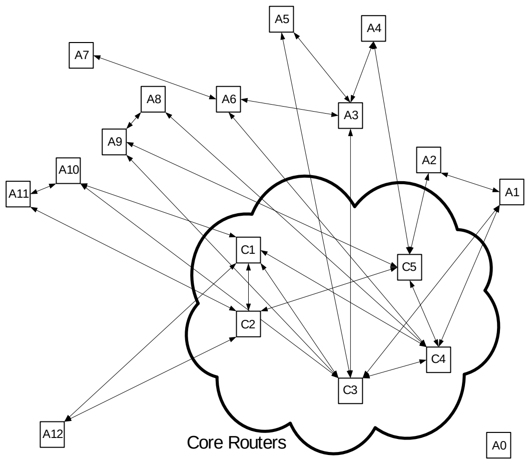

Fig. 12. Example isolated network

Table 11. Link reliabilities

Link	Reliability
Core–Core	0.9999
Access–Core	0.999
Access–Access	0.999
Wireless links	0.99

Since there are 18 nodes, there are ${}_{18}C_2$ = 153 ways to form a link. Hence there are ${}_{153}C_2$=11628 ways to add 2 different links. In order to reconnect A0 to the network, at least one of the links must attach to A0. Therefore, those configurations for which both links are not attached to A0 can be ruled out, and this leaves ${}_{153}C_2 - {}_{136}C_2$ = 2448 valid configurations to choose from.

We applied the SCR scheme to the 2448 candidate networks using 100,000 samples, it took 341 seconds on a 2.8GHz Pentium 4 machine to find the optimal network as shown in Fig. 13. The optimal network has a failure probability of 1.2485×10^{-4}. Intuitively, one might place the two links connecting A0 to its nearest core routers C3 and C4 as shown in Fig. 14. However, the intuitive network has a failure probability of 1.1069×10^{-3}, almost nine times that of the optimal network. It shows that spending a little time on searching for the optimal configuration can have significant benefits.

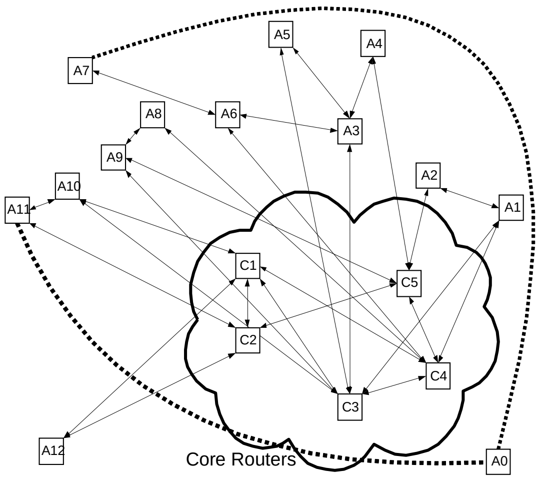

Fig. 13. Optimal network with two added links

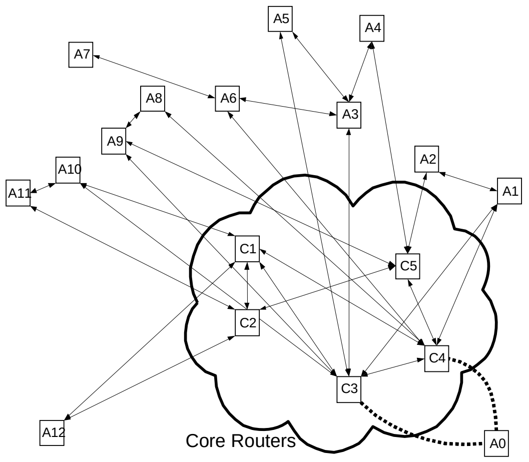

Fig. 14. Intuitive way to add two links

6.3 CE Method

Direct reliability comparison using the SCR scheme is effective when the number of candidates is small, however, it not practical to compare large number of networks even with the efficient SCR scheme. For example, there are 585,276 candidate networks if we want to add three links to the network in Figure 12. It quickly grows to nearly 22 million candidates if four links are to be added. Obviously we need a different approach to the problem and the CE-method is a good one.

Random Network Generation

The first step in the CE method is to generate random network according to some random mechanism. One such mechanism involves drawing without replacement. Imagine that each additional link is present in a lucky draw barrel from which we draw a fixed number of links to build our network. Initially, each link has an equal weight/probability of being picked. As the CE method progresses, the selection probabilities of the links are being modified until each one is close to either 0 or 1.

Let $\mathbf{w} = (w_1,\ldots,w_m)$ denote the weight vector, with $w_e \in [0,1]$ being the weight of edge e. If B represents the set of edges still in the barrel, then the probability of edge i being picked is

$$\frac{w_i}{\sum_{e\in B} w_e}.$$

The edges are drawn without replacement until the required number of edges is reached. The weights are updated at the end of each iteration of the CE method. Each weight w_e is increased if edge e is more likely to be involved in the high scoring networks or decreased otherwise. At the end of the CE optimization, w_e will be close to either 1 (part of the optimal network) or 0 (not part of the optimal network).

Updating Generation Parameters

The second part of the CE-method concerns updating the random network generation parameters w_e. It involves taking the best performing (e.g. 5%) random networks generated and finding which links are more likely being involved.

To find the elite portion of the candidate networks, one can extend the SCR scheme to search the top ρ portion instead of just the best network. For each permutation π, order the K candidate networks in ascending critical number. Then find the $\lceil \rho \times K \rceil$-th number and call it the elite critical number $\mathrm{b}^{\rho}(\pi)$. With N random permutations, we can estimate the probability of network $\mathcal{G}_i$ in the elite ρ portion by

$$P_\rho\left(\mathcal{G}_i\right) = \sum_{j=1}^{N} \frac{I\left\{\mathrm{b}_i\left(\Pi_{(j)}\right) \le \mathrm{b}^{\rho}\left(\Pi_{(j)}\right)\right\}}{N}.$$

The elite network set $\mathcal{G}^{\rho}$ consists of the $\lceil \rho \times K \rceil$ networks that have the highest $P_\rho(\mathcal{G}_i)$. Once we have the elite network set, we can update each edge weight by finding the probability of the edge being used in the elite networks, that is,

$$w_e' = \sum_{\mathcal{G}_i \in \mathcal{G}^{\rho}} \frac{I\left\{e \in \mathcal{G}_i\right\}}{\left|\mathcal{G}^{\rho}\right|}.$$

It is often beneficial to “smooth” the parameter update by incorporating part of the past history, especially when dealing with a noisy optimization problem. Let $\mathbf{w}_t$ be the weight vector used in the t-th iteration of the CE-method. A smoothing parameter $\alpha \in [0,1]$ is used to update the weight for the next iteration:

$$\mathbf{w}_{t+1} = \alpha \mathbf{w}_t + (1-\alpha)\mathbf{w}_t'.$$

Putting the sample generation and updating together, the CE-method algorithm can be summarized as follows:

Algorithm 3 (Simple CE Algorithm)

1. **Initialization**. S*et all* edge weight to equal value $w_{0,e} = 0.5$.
2. **Generation**. Generate K (e.g. 1000) random networks by drawing m_a additional edges from the candidate edges without replacement.
3. **Elite Networks**. Rank the random networks using the SCR scheme to find the best ρ portion (e.g. 5%) for edge weight update.
4. **Updating**. Update the edge weight by

$$w_{t+1,e} = \alpha w_{t,e} + (1-\alpha) \sum_{\mathcal{G}_i \in \mathcal{G}^\rho} \frac{I\{e \in \mathcal{G}_i\}}{|\mathcal{G}^\rho|}. \tag{26}$$

5. **Termination**. Repeat from Step 2 until every element in $\mathbf{w}_t$ lies in the ranges [0,0.01] or [0.99,1], say. The m_a edges with $w_e \in [0.99,1]$ are the optimal edges to be added.

Example 11

In this example, three links are to be added to the base network in Fig. 12 and the result is shown in Fig. 15.

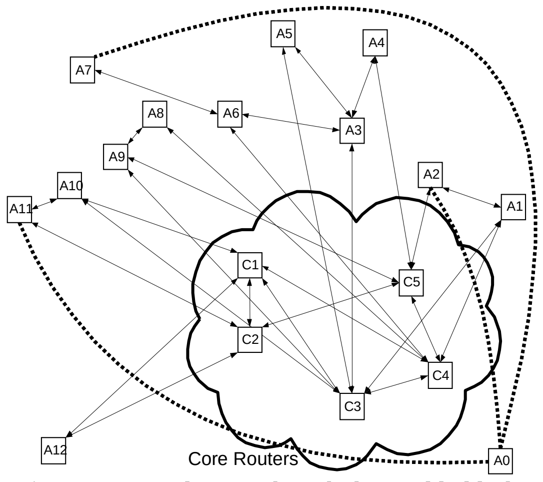

Fig. 15. Optimal network with three added links

In each iteration, K = 1000 random networks are generated and ρ = 5% or 50 elite networks are used to update the edge weights. The smoothing parameter of α = 0.5 is used and 5000 random permutations are used to find the elite network set. It took 19 iterations and 142 seconds on a 2.8GHz Pentium 4 machine to find the optimal configuration. Fig. 16 shows how the edge weights evolve over the iterations. It shows how the weights of the prospective links grow while the others decay toward zero. Eventually, the weights become “degenerate”, so that the edge weights of the optimal edge set stay at 1 while the remaining ones stay at 0.

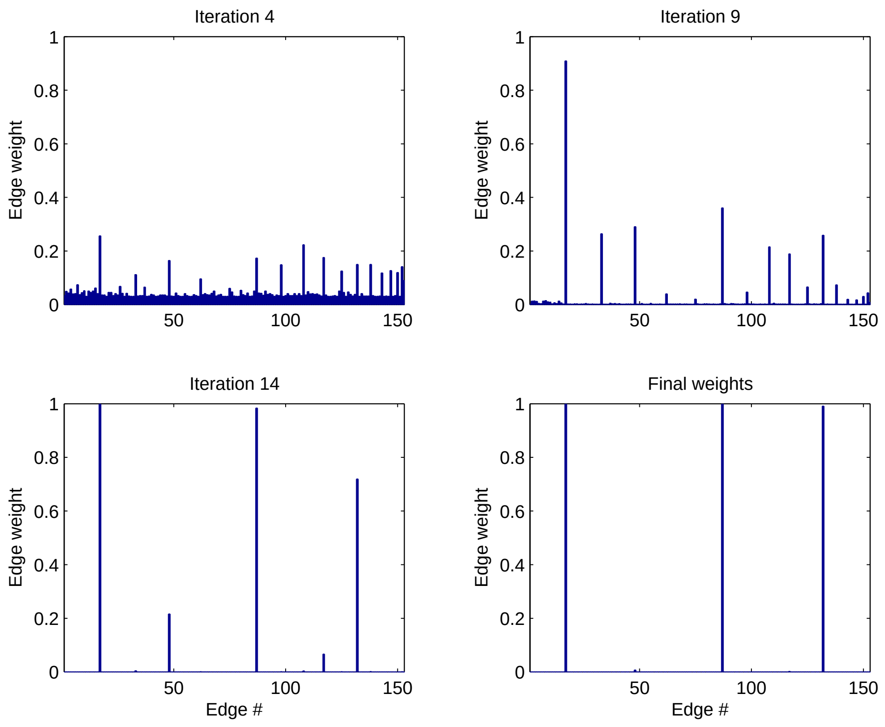

Fig. 16. Evolution of edge weights

This program was also used to find the optimal configuration for the two added links case. The same configuration as in Fig. 13 was found in 15 iterations using 105 seconds on the same machine using the same CE parameters. It demonstrates that the CE-method is an efficient and effective approach to finding the optimal configuration.

6.4 Hybrid Optimization Method

Algorithm 3 described above is tailored to finding a single optimal solution. If there is more than one solution, the algorithm has trouble deciding which one is better and as a consequence the weights “oscillate” and this keeps the algorithm from converging. Another situation where this occurs is when there are networks with performances that are very close to the optimal one.

Consider for instance in the previous examples the case where only one link is to be added to reconnect the network. There are 17 ways to reconnect node A0 to the rest of the network and each resultant network has exactly the same reliability. In this multiple optima situation, the Simple CE algorithm above will not converge.

Multi-optima Termination

One way to avoid the non-converging situation in the Simple CE algorithm is to terminate the algorithm once oscillating behavior is detected. In the case where the candidate networks have the same reliability and are perfectly matched, the SCP ranking scheme can detect the non-converging situation effectively, unlike independent sampling schemes, which will have real trouble telling whether two networks have the same reliability. If each of the N (e.g. 5000) samples indicates that every network comes up at exactly the same point, it is quite certain that all the K networks indeed have the same reliability. In that case the CE-method can stop and take the distinct networks as multiple optimal solutions.

Take the single additional link example described above using $K = 1000$ networks in each iteration and take $N = 5000$ samples to estimate the ranking. After 18 iterations that took 97 seconds, all the networks appeared to have the same reliability and the program finished. Among the 1000 generated networks, only 17 distinct networks exist and they are the 17 possible optimal solutions.

Mode Switching

In the situation where only one single optimal solution exists but there are other networks with reliabilities very close to the optimal, the Multi-optima Termination method may not work. One may try to terminate a prolonged simulation by detecting edge weight fluctuations, but unfortunately sometimes edge weight fluctuation before converging is part of the normal process. Therefore it is not easy to decide how long one should wait before the simulation is deemed to be non-converging. Even if it can be detected, it still does not help in searching for the optimal solution.

A more practical approach is to stop the CE iterations once the number of prospective links drops below a certain threshold, and then generate all the candidate networks using the prospective links and search for the optimal network using the SCR scheme. Since the edge weights will be polarized (close to either 0 or 1) in the CE-method, prospective links are simply those that have higher than the mean weight. Effectively, the CE-method is used as a filter removing those edges that are unlikely to be part of the optimal network.

To demonstrate how the scheme works, we repeat the two additional link examples with a much smaller sample size. Instead of 5000 samples, 100 were used to rank the 1000 generated networks. With such small number of samples, the confidence of the ranks is much reduced or, in other words, there is much more noise in the elite estimates. There is a high chance that the confidence intervals of top few candidate networks overlap each other. Therefore the algorithm is more likely to oscillate or even pick a sub-optimal solution if let to run indefinitely. However, we set the program to switch to SCR scheme when the number of candidate links drop below 15. Then 100,000 samples are used to search for the optimal configuration. The algorithm took eight iterations to filter out 140 of the 153 possible links and overall took 17 seconds to decide the optimal configuration is to add (A0, A11) and (A0, A7) links as shown in Fig. 13. Compare to 341 seconds required by the SCR algorithm alone, this hybrid scheme is much more robust and efficient. We applied the same scheme to the three additional link example with the switch-over point set to 10 links or below. The simulation took 21 seconds to find the same optimal solution depicted in Fig. 15.

To summarize, the Hybrid procedure is as follows:

Algorithm 4 (Hybrid CE Algorithm)

1. **Initialization.** Set all edge weight to an equal value $w_{0,e} = 0.5$.
2. **Generation.** Generate K (e.g. 1000) random networks by drawing m_a additional edges from the candidate edges without replacement.
3. **Elite Networks.** Rank the random networks using the SCR scheme to find the best ρ portion (e.g. 5%) for edge weight update.
4. **Multi-optima Condition.** Check if all generated networks have identical reliability. If yes, output distinct networks and terminate procedure.
5. **Mode Switching.** If the number of prospective links drops to or below a threshold m_p, generate all candidate network using the prospective links and use SCR to search for the optimal network. Output the optimal network and terminate the procedure.
6. **Updating.** Update the edge weight using Equation (26).
7. **Termination.** Repeat from Step 2 until every element in $\mathbf{w}_t$ lies in the ranges [0,0.01] or [0.99,1]. The m_a edges with w_e in the [0.99,1] range are the optimal edge set to be added.

6.5 Comparison Between the Methods

To compare the different schemes, the examples of 1, 2 and 3 additional links are re-run with more comparable parameters. For the SCR scheme, 500,000 samples are used to compare different networks. In the CE-methods, each iteration uses 10,000 samples to compare 1000 generated networks. The elite portion is set to 5% and a smoothing factor of 0.5 is used as well. With the Hybrid CE algorithm, the CE parameters are the same as for the Simple CE except a sample size of 1000 is used. The maximum prospective link is set so that the prospective network count is no more than 100. Once the threshold is reached, the prospective networks are generated for a final comparison using SCR with 500,000 samples. The average run-time in seconds are tabulated in Table 12.

Table 12. Comparison of different schemes

	1-Link	2-Link	3-Link
SCR scheme	16s	1572s	n/a
Simple CE	non-converging	234s	288s
Hybrid CE	18s	56s	53s

In the 1-Link case, which has 17 multiple optimal solutions, the Simple CE algorithm does not converge and will run indefinitely if allowed. The SCR scheme quickly determined that the 17 candidate networks have the same reliability. Note that in this case the SCR scheme is a semi-automatic process. It requires manual filtering of the candidate networks that will not reconnect the node A0. If all possible candidates were used in the comparison, it would take 102 seconds. With the Hybrid CE algorithm, it took only 2 seconds to filter out the 136 candidates and another 16 seconds to find out that the remaining 17 networks have the same reliability. It is a fully automatic process without the need of manual filtering.

In the 2-Link case, there are 2448 networks to compare for the SCR scheme and it takes a fairly long time (over 25 minutes) to finish. On the other hand, the Simple CE algorithm converges to the optimal solution in less than 4 minutes. The Hybrid CE algorithm is able to cut the computation time by 75% to less than 1 minute. This is achieved by switching about half-way during the iterations and uses the SCR scheme to compare a small number of prospective candidates.

In the 3-Link case, the SCR method was not performed because it would have taken too long. Since the number of valid network grows exponentially with the added links, the 3-Link example is projected to take over 30 hours to compute using the SCR scheme. The Simple CE and the Hybrid CE algorithms behave similar to the 2-Link case. It is interesting to note that the Hybrid CE algorithm is faster in the 3-Link case than the 2-Link case. This is due to quantization of combinations: the three highest 2-Link combinations under 100 are {91, 78, 55} while that of the 3-Link case are {84, 56, 35}. In fact at the point of switch over, the 2-Link case has an average of 67 networks to compare while the 3-Link case has 48.

Acknowledgements

We thank Sho Nariai for providing numerical results for the network planning problem. The first author acknowledges the financial support of the Australian Research Council, under grant number DP0558957.

References

1. R.E. Barlow and F. Proschan. *Statistical Theory of Reliability and Life Testing*. Holt, Rinehart and Wilson., 1975.
2. A.T. Bharucha-Reid. *Elements of the Theory of Markov Processes and Their Applications*. McGraw-Hill, 1960.
3. Yu Burtin and B. Pittel. Asymptotic estimates of the reliability of a complex system. *Engrg. Cybern.*, 10(3):445 – 451, 1972.
4. H. Cancela and M.E. Urquhart. Simulated annealing for communication network reliability improvement. In *Proceedings of the XXI Latin American Conference on Informatics*. CLEI-SBC, July 1995.
5. S.X. Chen and J.S. Liu. Statistical Applications of the Poisson-Binomial and Conditional Bernoulli Distributions. *Statistical Sinica*, 7:875–892, 1997
6. C.J. Colbourn. *The Combinatorics of Network Reliability*. Oxford University Press, 1987.
7. C.J. Colbourn and D.D. Harms. Evaluating performability: Most probable states and bounds. *Telecommunication Systems*, 2:275 –300, 1994.
8. P.-T. de Boer, D.P. Kroese, S. Mannor, and R. Y. Rubinstein. A tutorial on the cross-entropy method. *Annals of Operations Research*, 134(1):19 –67, 2005.
9. B. Dengiz, F. Altiparmak, and A. E. Smith. Local search genetic algorithm for optimal design of reliable networks. *IEEE Transactions on Evolutionary Computation*, 1(3):179 –188, September 1997.
10. M.C. Easton and C.K. Wong. Sequential destruction method for Monte Carlo evaluation of system reliability. *IEEE Transactions on Reliability*, R-29:27 –32, 1980.
11. T. Elperin, I.B. Gertsbakh, and M. Lomonosov. Estimation of network reliability using graph evolution models. *IEEE Transactions on Reliability*, 40(5):572 –581, 1991.

12. T. Elperin, I.B. Gertsbakh, and M. Lomonosov. An evolution model for Monte Carlo estimation of equilibrium network renewal parameters. *Probability in the Engineering and Informational Sciences*, 6:457 –469, 1992.
13. G.S. Fishman. A Monte Carlo sampling plan for estimating network reliability. *Operation Research*, 34(4):122 –125, 1986.
14. I.B. Gertsbakh. *Reliability Theory with Application to Preventive Maintenance*. Springer, 2000.
15. I.B. Gertsbakh. *Statistical Reliability Theory*. Marcel Dekker, 1989.
16. I.I. Gikhman and A. V. Skorokhod. *Introduction to the Theory of Random Processes*. Dover Publications, 1996.
17. K-P. Hui, N. Bean, and M. Kraetzl. Network reliability difference estimation. *submitted for publication*, 2005.
18. K-P. Hui, N. Bean, M. Kraetzl, and D.P. Kroese. The tree cut and merge algorithm for estimation of network reliability. *Probability in the Engineering and Informational Sciences*, 17(1):24 –45, 2003.
19. K-P. Hui, N. Bean, M. Kraetzl, and D.P. Kroese. The Cross-Entropy Method for Network Reliability Estimation. *The Annals of Operations Research*, 134:101 –118, 2005.
20. H. Kumamoto, K. Tanaka, K. Inoue, and E. J. Henley. Dagger sampling Monte Carlo for system unavailability evaluation. *IEEE Transactions on Reliability*, R-29(2):376 –380, 1980.
21. M. Lomonosov. An Evolution Model for Monte Carlo Estimation of Equilibrium Network Renewal Parameters. *Probability in the Engineering and Informational Sciences*, 6:457 – 469, 1992.
22. M. Lomonosov. On Monte Carlo estimates in network reliability. *Probability in the Engineering and Informational Sciences*, 8:245 –264, 1994.
23. J.R. Norris. *Markov Chains*. Cambridge University Press, 1997.
24. J.S. Provan and M.O. Ball. The complexity of counting cuts and of computing the probability that a graph is connected. *SIAM Journal of Computing*, 12:777 –787, 1982.
25. D. Reichelt, F. Rothlauf, and P. Bmilkowsky. Designing reliable communication networks with a genetic algorithm using a repair heuristic. In *Evolutionary Computation in Combinatorial Optimization*. Springer-Verlag Heidelberg, 2004.
26. R.Y. Rubinstein. Optimization of computer simulation models with rare events. *European Journal of Operations Research*, 99:89 –112, 1997.
27. R.Y. Rubinstein. The cross-entropy method for combinatorial and continuous optimization. *Methodology and Computing in Applied Probability*, 2:127 – 190, 1999.
28. R.Y. Rubinstein and D. P. Kroese. *The Cross-Entropy Method: A unified approach to Combinatorial Optimization, Monte Carlo Simulation and Machine Learning*. Springer-Verlag, New York, 2004.
29. R.Y. Rubinstein. Combinatorial optimization, cross entropy, ants and rare events. *Stochastic Optimization: Algorithms and Applications*, pages 303 – 363, 2001.

30. R.Y. Rubinstein and B. Melamed. *Modern simulation and modeling*. Wiley series in probability and Statistics, 1998.
31. C. Srivaree-Ratana and A. E. Smith. Estimating all-terminal network reliability using a neural network. *Proceedings of the 1998 IEEE International Conference on Systems, Man, and Cybernetics*, 5:4734 –4740, October 1998.
32. W.-C. Yeh. A new Monte Carlo method for the network reliability. *Proceedings of First International Conference on Information Technologies and Applications(ICITA2002)*, November 2002.
33. Y-S. Yeh, C. C. Chiu, and R-S. Chen. A genetic algorithm for *k*-node set reliability optimization with capacity constraint of a distributed system. *Proc. Natl. Sci. Counc. ROC(A)*, 25(1):27 –34, 2001.