Dept. of Math. University of Oslo
Statistical Research Report No. 2
ISSN 0806–3842 January 2004

SYSTEM RELIABILITY EVALUATION USING CONDITIONAL MONTE CARLO METHODS

Arne Bang Huseby, Morten Naustdal, Ingeborg Drengstig Vårli

Abstract

The paper shows how Monte Carlo methods can be improved significantly by conditioning on a suitable variable or vector. In particular this principle is applied to system reliability evaluation. Different choices of variables to condition on lead to different approaches. We start out by using upper and lower bounds on the structure function of the system, and develop an efficient method for sampling from the resulting conditional distribution. Another approach is to use the sum of the component state variables. In relation to this an efficient algorithm for simulating a vector of independent Bernoulli variables given their sum is presented. By using this algorithm one can generate such a vector in $O(n)$ time, where n is the number of variables. Thus, simulating from the conditional distribution can be done just as efficient as simulating from the unconditional distribution. The special case where the Bernoulli variables are i.i.d. is also considered. For this case the reliability evaluation can be improved even further. In particular, we present a simulation algorithm which enables us to estimate the entire system reliability polynomial expressed as a function of the common component reliability. If the component reliabilities are not too different from each other, a generalized version of the improved conditional method can be used in combination with importance sampling. Finally we outline how the two conditioning methods can be combined in order to get even better results.

1 INTRODUCTION

As a result of the availability of high-speed computers, the use of Monte Carlo methods have accelerated. In many cases, however, it is necessary to use various clever tricks in order to make the estimates converge faster. In particular this is true in situations where one needs to estimate something that involves events with very low probabilities. One such area is system reliability estimation. Since failure events often have very low probability, a large number of simulations is needed in order to obtain stable results. Sometimes, however, conditioning can be used to improve the convergence.

In the general case the principle of conditioning can be stated as follows: Let $\boldsymbol{X} = (X_n, \dots, X_n)$ be a random vector with a known distribution, and assume that we want to calculate the expected value of $\phi = \phi(\boldsymbol{X})$. We denote this expectation by h. Assume furthermore that the distribution of ϕ cannot be derived analytically in polynomial time with respect to n. Using Monte Carlo simulations, however, we can proceed by generating a sample of size N, of independent vectors $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N$, all having the same distribution as $\boldsymbol{X}$, and then estimate h with the simple *Monte Carlo* estimate:

$$\hat{h}_{MC} = \frac{1}{N}\sum_{r=1}^{N} \phi(\boldsymbol{X}_r), \qquad \text{where} \qquad \mathrm{Var}\left(\hat{h}_{MC}\right) = \mathrm{Var}(\phi)/N \tag{1.1}$$

Now, assume that $S = S(\boldsymbol{X})$ is some other function whose distribution can be calculated analytically in polynomial time w.r.t. n. More specifically, we assume that S is a *discrete* variable with values in the set $\{s_1, \ldots, s_k\}$. We also introduce:

$$\theta_j = E[\phi \mid S = s_j], \qquad j = 1, \ldots, k\ . \tag{1.2}$$

We then have:

$$h = \sum_{j=1}^{k} \theta_j \Pr(S = s_j) \tag{1.3}$$

Instead of generating N samples from the distribution of $\boldsymbol{X}$as in (1.1), we divide the set into k groups, one for each θ_j, where the j-th group has size N_j, $j = 1, \ldots, k$, and $N_1 + \cdots + N_k = N$. The data in the j-th group are sampled from the conditional distribution of $\boldsymbol{X}$ given that $S = s_j$, and are used to estimate the conditional expectation θ_j, $j = 1, \ldots, k$. Denoting the data in the j-th group by $\{\boldsymbol{X}_{r,j} : r = 1, \ldots, N_j\}$, θ_j is estimated by:

$$\hat{\theta}_j = \frac{1}{N_j}\sum_{r=1}^{N_j} \phi(\boldsymbol{X}_{r,j}), \qquad j = 1, \ldots, k\ . \tag{1.4}$$

By combining these estimates, we get the *conditional Monte Carlo estimate*:

$$\hat{h}_{CMC} = \sum_{j=1}^{k} \hat{\theta}_j \Pr(S = s_j), \tag{1.5}$$

where:

$$\mathrm{Var}(\hat{h}_{CMC}) = \sum_{j=1}^{k} \mathrm{Var}(\phi \mid S = s_j)[\Pr(S = s_j)]^2/N_j.$$

We observe that the variance of the conditional estimate depends on the choices of the N_j's. Ideally we would like N_j to be large if the product of the conditional variance and the squared probability is large, and small otherwise, as this would yield the most efficient partition of the total sample with respect to minimizing the total variance. In practice, however, this is difficult, since the conditional variance is typically not known. In order to compare the result with the variance of the original Monte Carlo estimate, we let $Nj \approx N \cdot \Pr(S = s_j)$, $j = 1, \ldots, k$. With this choice we get:

$$\begin{aligned} Var(\hat{h}_{CMC}) &\approx \sum_{j=1}^{k} \mathrm{Var}(\phi \mid S = s_j) \Pr(S = s_j)/N \qquad (1.6) \\ &= E[\mathrm{Var}(\phi \mid S)]/N \\ &= \{\mathrm{Var}(\phi) - \mathrm{Var}[E[\phi \mid S)]\}/N \leq \mathrm{Var}(\phi)/N = \mathrm{Var}(\hat{h}_{MC}). \end{aligned}$$

From the above equation we see that the conditional estimate has smaller variance than the original Monte Carlo estimate provided that $\mathrm{Var}[E(\phi \mid S)]$ is positive. This quantity can be interpreted as a measure of how much information S contains relative to ϕ. Thus, when looking for good choices for S we should look for variables containing as much information about ϕ as possible. However, there are some other important points that need to be considered. First of all S must have a distribution that can be derived analytically in polynomial time. Next, the number of possible values of S, i.e., k, must be polynomially limited by n. Finally, it must be possible to sample efficiently from the distribution of $\boldsymbol{X}$ given S.

The problem of sampling from conditional distributions has been studied recently by several authors. Lindqvist and Taraldsen[11] proposes a general method for sampling from conditional distributions in cases where S is a sufficient statistic. Of special relevance to the present paper, is Broström and Nilsson[2] who consider the problem of sampling independent Bernoulli variables given their sum. See also Nilsson[13]. In the remaining part of this paper we shall focus on applications in reliability theory, where $\boldsymbol{X}$ typically is a vector of independent Bernoulli variables. In relation to this we shall derive our own conditional sampling methods.

We now apply the above ideas in the context of system reliability. That is, we assume that $\boldsymbol{X}$ is a vector of independent Bernoulli variables, interpreted as the component state vector relative to a system of n components. A component is *failed* if its state variable is 0, and *functioning* if the state variable is 1. The function ϕ is also assumed to take values 0, 1, and interpreted as the *system state*. If ϕ is 0, the system is *failed*, and if ϕ is 1, the system is *functioning*. The function ϕ is referred to as the structure function of the system, and is assumed to be a nondecreasing function of the component state vector, $\boldsymbol{X}$. In general the calculation of ϕ for a given component state vector depends very much on the representation of this function. If e.g., the system is represented as some sort of communication network its state can usually be evaluated in $O(n)$ time or better. If on the other hand the system is specified in terms of a list of minimal path (or cut) sets, the evaluation can be very slow since the number of such sets in the worst cases grows exponentially with the number of components. For this study, however, we assume that ϕ can be calculated in $O(n)$ time for a given component state vector.

The expectation h is called the *system reliability*, and is of course equal to $\Pr(\phi = 1)$. For an introduction to reliability theory, we refer to Barlow and Proschan[1].

There are of course many different choices for the variable S which can be used in a reliability setting. In the next two sections we will consider two possible such choices.

2 Conditioning on upper and lower bounds on the structure function

In this section, which is based on the results of Vårli[15], we let $S = (\phi_L, \phi_U)$, where ϕ_L and ϕ_U are the structure functions of two simpler systems such that $\phi_L \leq \phi \leq \phi_U$. If these two functions are close approximations to ϕ, a lot can be gained. For similar approaches, see Cancela and Khadari[3], [4], [5] and Fishman[6], [7].

With this choice of S the set of possible values of S is $\{(0,0),(0,1),(1,1)\}$. Moreover, $E[\phi \mid \phi_L = \phi_U = 0] = 0$ and $E[\phi \mid \phi_L = \phi_U = 1] = 1$. Thus, the only quantity we need to estimate is $E[\phi \mid \phi_L = 0, \phi_U = 1]$. We also observe that:

$$\begin{aligned} \Pr(S=(0,1)) &= \Pr(\phi_L = 0 \cap \phi_U = 1) \\ &= \Pr(\phi_U = 1) - \Pr(\phi_L = 1) = E[\phi_U] - E[\phi_L] = h_U - h_L, \end{aligned} \tag{2.1}$$

where h_U and h_L denote the reliabilities of the upper and lower bound systems respectively. Similarly, we have that:

$$\Pr(S=(1,1)) = \Pr(\phi_L = 1 \cap \phi_U = 1) = \Pr(\phi_L = 1) = E[\phi_L] = h_L. \tag{2.2}$$

By inserting these relations into (1.5), we see that in this case the conditional Monte Carlo estimate takes the following form:

$$\hat{h}_{CMC} = (h_U - h_L)\frac{1}{N}\sum_{r=1}^{N} \phi(\boldsymbol{X}_r) + h_L, \tag{2.3}$$

where $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N$ are sampled from the conditional distribution of $\boldsymbol{X}$ given that $\phi_L = 0$ and $\phi_U = 1$. Denoting this distribution by Q, we obtain the following expression for the variance of the estimate:

$$\begin{aligned} \operatorname{Var}_Q(\hat{h}_{CMC}) &= \frac{1}{N}(h_U - h_L)^2 \operatorname{Var}_Q[\phi(\boldsymbol{X})] \\ &= \frac{1}{N}(h_U - h_L)^2 [E_Q\phi(\boldsymbol{X}) - (E_Q\phi(\boldsymbol{X}))^2] \\ &= \frac{1}{N}(h_U - h_L)^2 [\frac{h - h_L}{h_U - h_L} - (\frac{h - h_L}{h_U - h_L})^2] \\ &= \frac{1}{N}(h_U - h)(h - h_L) \end{aligned} \tag{2.4}$$

where E_Q and Var_Q denote expectation and variance with respect to the distribution Q. This variance gets its maximum value when h is equal to the mean of h_U and h_L, i.e., when $h = (h_U + h_L)/2$. Moreover, this maximum is given by:

$$\max\{\operatorname{Var}_Q\} = \frac{(h_U - h_L)^2}{4N} \tag{2.5}$$

From (2.5) we see that the best results are obtained when h_U and h_L are as close to each other as possible, which of course is intuitively obvious.

We now turn to the problem of sampling from the conditional distribution. Before we present this, we introduce the (unconditional) component reliabilities $p_i = \Pr(X_i = 1)$, $i = 1, \ldots, n$. We also need the following notation: $(\boldsymbol{x}_i, \boldsymbol{p}) = (x_1, \ldots, x_i, p_{i+1}, \ldots, p_n)$.

By writing the conditional distribution Q as:

$$\begin{aligned} &\Pr(X_1 = x_1, \ldots, X_n = x_n \mid \phi_U(\boldsymbol{X}) - \phi_L(\boldsymbol{X}) = 1) \\ &\quad = \prod_{i=1}^{n} \Pr(X_i = x_i \mid \phi_U(\boldsymbol{X}) - \phi_L(\boldsymbol{X}) = 1, \bigcap_{k=1}^{i-1} X_k = x_k), \end{aligned} \tag{2.6}$$

we see that the sampling problem is reduced to sampling each component state variable X_i from its corresponding conditional probability distribution relative to this expansion, i.e., by using $\Pr(X_i = x_i \mid \phi_U(\boldsymbol{X}) - \phi_L(\boldsymbol{X}) = 1, \bigcap_{k=1}^{i-1} X_k = x_k)$. Since the component state variables are assumed to be independent unconditionally, these probabilities can be calculated as follows:

$$\begin{aligned}
&\Pr(X_i = x_i \mid \phi_U(\boldsymbol{X}) - \phi_L(\boldsymbol{X}) = 1, \bigcap_{k=1}^{i-1} X_k = x_k) \qquad (2.7)\\
&= \frac{\Pr(\phi_U(\boldsymbol{X}) - \phi_L(\boldsymbol{X}) = 1 \mid \bigcap_{k=1}^{i} X_k = x_k)}{\Pr(\phi_U(\boldsymbol{X}) - \phi_L(\boldsymbol{X}) = 1 \mid \bigcap_{k=1}^{i-1} X_k = x_k)} \Pr(X_i = x_i \mid \bigcap_{k=1}^{i-1} X_k = x_k)\\
&= \frac{E(\phi_U \mid \bigcap_{k=1}^{i} X_k = x_k) - E(\phi_L \mid \bigcap_{k=1}^{i} X_k = x_k)}{E(\phi_U \mid \bigcap_{k=1}^{i-1} X_k = x_k) - E(\phi_L \mid \bigcap_{k=1}^{i-1} X_k = x_k)} \Pr(X_i = x_i)\\
&= \frac{h_U(\boldsymbol{x}_i, \boldsymbol{p}) - h_L(\boldsymbol{x}_i, \boldsymbol{p})}{h_U(\boldsymbol{x}_{i-1}, \boldsymbol{p}) - h_L(\boldsymbol{x}_{i-1}, \boldsymbol{p})} p_i^{x_i} (1-p_i)^{1-x_i}
\end{aligned}$$

From (2.7) it follows that if ϕ_L and ϕ_U are chosen so that their respective reliabilities can be calculated in polynomial time, then it is possible to sample from the conditional distribution Q in polynomial time as well. In the remaining part of this section we will present three different methods for obtaining such structure functions.

2.1 The disjoint path and cut sets method

This approach, introduced by Fishman[6], starts out by considering the collections of minimal path and cut sets of the system, which we denote respectively by $\mathcal{P}$ and $\mathcal{C}$. Both these collections uniquely determines the system, and hence can in principle be used to calculate the system reliability e.g., by applying the following well-known relations:

$$h = E[\coprod_{P \in \mathcal{P}} \prod_{i \in P} X_i] = E[\prod_{C \in \mathcal{C}} \coprod_{i \in C} X_i] \qquad (2.8)$$

In practice, however, evaluating this formula is difficult, partly because the path sets (or cut sets) typically are not pairwise disjoint, and partly because the number of path and cut sets in the system may in the worst cases grow exponentially with the number of components in the system.

Now, let $\mathcal{P}'$ and $\mathcal{C}'$ be subcollections of $\mathcal{P}$ and $\mathcal{C}$ respectively, and assume that both these subcollections consist only of pairwise disjoint sets. It is of course always possible to construct such subcollections by leaving out sufficiently many of the sets in the original collections. In fact it is usually possible to generate such subcollections directly without generating the complete collections of minimal path or cut sets first. Moreover, the number of sets in $\mathcal{P}'$ is always less than or equal to the number of components in the system. The same holds true for $\mathcal{C}'$ as well. Thus, the sizes of these subcollections are relatively small.

We then define ϕ_L and ϕ_U as follows:

$$\begin{aligned}
\phi_L(\boldsymbol{X}) &= \coprod_{P \in \mathcal{P}'} \prod_{i \in P} X_i, \qquad (2.9)\\
\phi_U(\boldsymbol{X}) &= \prod_{C \in \mathcal{C}'} \coprod_{i \in C} X_i.
\end{aligned}$$

The lower bound structure, ϕ_L, is a type of structure which we will refer to as *a disjoint path sets structure*, i.e., a collection of disjoint series structures organized in a parallel structure (see Figure 1). Similarly, the upper bound structure, ϕ_U, is a type of structure which we will refer to as *a disjoint cut sets structure*, i.e., a collection of disjoint parallel structures organized in a series structure (see Figure 2).

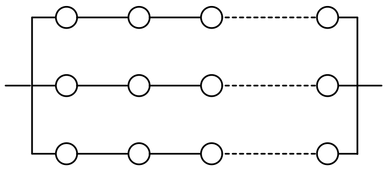

Figure 1: A disjoint paths structure.

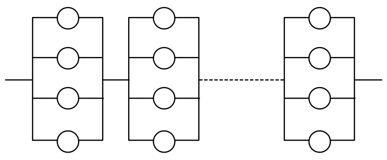

Figure 2: A disjoint cuts structure.

Since $\mathcal{P}' \subseteq \mathcal{P}$ and $\mathcal{C}' \subseteq \mathcal{C}$, it follows that $\phi_L \leq \phi \leq \phi_U$. Moreover, by using that the subcollections consist only of pairwise disjoint sets, the reliabilities of ϕ_L and ϕ_U can be calculated in $O(n)$ time by using the following formulas:

$$\begin{aligned} h_L(\boldsymbol{p}) &= \coprod_{P \in \mathcal{P}'} \prod_{k \in P} p_k, \\ h_U(\boldsymbol{p}) &= \prod_{C \in \mathcal{C}'} \coprod_{k \in C} p_k. \end{aligned} \tag{2.10}$$

Hence, all the conditional expectations needed in the distribution Q can be calculated for $r = i-1, i$ as follows:

$$E(\phi_L \mid \bigcap_{k=1}^{r} X_k = x_k) = h_L(\boldsymbol{x}_r, \boldsymbol{p}) = \coprod_{P \in \mathcal{P}'} [\prod_{k \in P \setminus E_r} p_k][\prod_{k \in P \bigcap E_r} x_k] \tag{2.11}$$

$$E(\phi_U \mid \bigcap_{k=1}^{r} X_k = x_k) = h_U(\boldsymbol{x}_r, \boldsymbol{p}) = \prod_{C \in \mathcal{C}'} [\coprod_{k \in C \setminus E_r} p_k][\coprod_{k \in C \bigcap E_r} x_k], \tag{2.12}$$

where we define $E_r = \{1, \ldots, r\}$, $r = 1, \ldots, n$.

Note that since ϕ_L and ϕ_U are constructed from subcollections of $\mathcal{P}$ and $\mathcal{C}$, it may happen that some of the components are irrelevant w.r.t. these systems. If i is irrelevant w.r.t both systems, it follows that:

$$\begin{aligned} h_U(\boldsymbol{x}_i, \boldsymbol{p}) &= h_U(\boldsymbol{x}_{i-1}, \boldsymbol{p}), \\ h_L(\boldsymbol{x}_i, \boldsymbol{p}) &= h_L(\boldsymbol{x}_{i-1}, \boldsymbol{p}). \end{aligned} \tag{2.13}$$

Thus for this component we get that:

$$\frac{h_U(\boldsymbol{x}_i, \boldsymbol{p}) - h_L(\boldsymbol{x}_i, \boldsymbol{p})}{h_U(\boldsymbol{x}_{i-1}, \boldsymbol{p}) - h_L(\boldsymbol{x}_{i-1}, \boldsymbol{p})} = 1 \tag{2.14}$$

By inserting (2.14) into (2.7), we see that the resulting sampling probability becomes equal to the unconditional probability. Thus, for components which are irrelevant w.r.t. to the upper and lower bound structures, the sampling procedure is slightly simplified since we do not have to calculate any of the values of h_L or h_U. Still for the remaining set of components, these quantities need to be computed. However, as pointed out in Fishman[6], it is possible to carry out these computations sequentially so that one conditional probability is calculated from the previous one by a simple updating scheme. In order to explain this in more details we introduce the following notation:

$$\begin{aligned}
\rho_P(\boldsymbol{x}_r, \boldsymbol{p}) &= [\prod_{k \in P \setminus E_r} p_k][\prod_{k \in P \bigcap E_r} x_k], \qquad P \in \mathcal{P}', r = 1, \ldots, n, \qquad (2.15)\\
\kappa_C(\boldsymbol{x}_r, \boldsymbol{p}) &= [\coprod_{k \in C \setminus E_r} p_k][\coprod_{k \in C \bigcap E_r} x_k], \qquad C \in \mathcal{C}', r = 1, \ldots, n.
\end{aligned}$$

Using this notation we can rewrite $h_L(\boldsymbol{x}_r, \boldsymbol{p})$ and $h_U(\boldsymbol{x}_r, \boldsymbol{p})$ as:

$$\begin{aligned}
h_L(\boldsymbol{x}_r, \boldsymbol{p}) &= \coprod_{P \in \mathcal{P}'} \rho_P(\boldsymbol{x}_r, \boldsymbol{p}), \qquad (2.16)\\
h_U(\boldsymbol{x}_r, \boldsymbol{p}) &= \prod_{C \in \mathcal{C}'} \kappa_C(\boldsymbol{x}_r, \boldsymbol{p}).
\end{aligned}$$

We then assume that we have calculated $h_U(\boldsymbol{x}_{i-1}, \boldsymbol{p})$ and $h_L(\boldsymbol{x}_{i-1}, \boldsymbol{p})$, and that we want to calculate $h_U(\boldsymbol{x}_i, \boldsymbol{p})$ and $h_L(\boldsymbol{x}_i, \boldsymbol{p})$. Since the sets in $\mathcal{P}'$ are disjoint, there exists at most one set in $\mathcal{P}'$, say P^*, such that $i \in P^*$. Similarly there exists at most one set in $\mathcal{C}'$, say C^*, such that $i \in C^*$. Then by (2.15) it follows that:

$$\begin{aligned}
\rho_{P^*}(\boldsymbol{x}_i, \boldsymbol{p}) &= \frac{x_i}{p_i} \rho_{P^*}(\boldsymbol{x}_{i-1}, \boldsymbol{p}), \qquad (2.17)\\
\kappa_{C^*}(\boldsymbol{x}_i, \boldsymbol{p}) &= 1 - \frac{1 - x_i}{1 - p_i}[1 - \kappa_{C^*}(\boldsymbol{x}_{i-1}, \boldsymbol{p})].
\end{aligned}$$

For the other path and cut sets going from $i-1$ to i has no effect. That is we have:

$$\begin{aligned}
\rho_P(\boldsymbol{x}_i, \boldsymbol{p}) &= \rho_P(\boldsymbol{x}_{i-1}, \boldsymbol{p}), \qquad \forall P \in \mathcal{P}', P \neq P^*, \qquad (2.18)\\
\kappa_C(\boldsymbol{x}_i, \boldsymbol{p}) &= \kappa_C(\boldsymbol{x}_{i-1}, \boldsymbol{p}), \qquad \forall C \in \mathcal{C}', C \neq C^*.
\end{aligned}$$

By inserting (2.17) and (2.18) into (2.16) we finally get:

$$\begin{aligned}
h_L(\boldsymbol{x}_i, \boldsymbol{p}) &= 1 - \frac{1 - \rho_{P^*}(\boldsymbol{x}_i, \boldsymbol{p})}{1 - \rho_{P^*}(\boldsymbol{x}_{i-1}, \boldsymbol{p})}[1 - h_L(\boldsymbol{x}_{i-1}, \boldsymbol{p})], \qquad (2.19)\\
h_U(\boldsymbol{x}_i, \boldsymbol{p}) &= \frac{\kappa_{C^*}(\boldsymbol{x}_i, \boldsymbol{p})}{\kappa_{C^*}(\boldsymbol{x}_{i-1}, \boldsymbol{p})} h_U(\boldsymbol{x}_{i-1}, \boldsymbol{p}).
\end{aligned}$$

Summarizing this we see that all the calculations needed to arrive at $h_L(\boldsymbol{x}_i, \boldsymbol{p})$ and $h_U(\boldsymbol{x}_i, \boldsymbol{p})$ assuming that we have calculated $h_L(\boldsymbol{x}_{i-1}, \boldsymbol{p})$ and $h_U(\boldsymbol{x}_{i-1}, \boldsymbol{p})$

(and stored the values of both these functions as well as the corresponding values of all the ρ_P- and κ_C-functions), can be done in constant time by using the updating formulas (2.17) and (2.19). Thus, the sampling procedure using this method can be carried out in $O(n)$ time.

2.2 The k-out-of-n method

This approach is based on using k-out-of-n structures as upper and lower bounds on the system. Thus, we assume as before that ϕ is a structure function of a system of n components, and introduce the size of the smallest path set of the system denoted by d, and the size of the smallest cut set denoted by c. We proceed by defining ϕ_L and ϕ_U as structure functions such that ϕ_L is an $(n-c+1)$-out-of-n system, and ϕ_U is a d-out-of-n system. From this it follows easily that $\phi_L \leq \phi \leq \phi_U$.

Before we calculate the sampling distribution Q, we introduce the following random variables:

$$S_m = \sum_{i=m}^{n} X_i \ , \qquad m = 1, \ldots, n \ . \tag{2.20}$$

The distributions of $S_1, \ldots, S_n$ can be calculated recursively using the following formula (modified in the obvious way in the limiting cases where $s = 0$ or $s = m$):

$$\Pr(S_m = s) = p_m \Pr(S_{m+1} = s-1) + (1-p_m) \Pr(S_{m+1} = s). \tag{2.21}$$

Thus, we start out by determining the distribution of S_n being equal to the distribution of X_n. We then proceed recursively by calculating the distribution of S_{n-1}, etc. A table containing all these distributions (including the distribution of $S = S_1$) can thus be derived in $O(n^2)$-time.

Using this all the conditional expectations needed in the distribution Q can be calculated for $r = i-1, i$ as follows by defining $z_r = \sum_{j=1}^{r} x_j$, $r = 1, \ldots, n$ that:

$$\begin{aligned} E(\phi_L \mid \bigcap_{k=1}^{r} X_k = x_k) &= \Pr(\sum_{k=1}^{n} X_k \geq n-c+1 \mid \bigcap_{k=1}^{r} X_k = x_k) \\ &= \Pr(\sum_{k=r+1}^{n} X_k \geq n-c+1-\sum_{j=1}^{r} x_j) \\ &= \Pr(S_{r+1} \geq n-c+1-z_r) \end{aligned} \tag{2.22}$$

$$\begin{aligned} E(\phi_U \mid \bigcap_{k=1}^{r} X_k = x_k) &= \Pr(\sum_{k=1}^{n} X_k \geq d \mid \bigcap_{k=1}^{r} X_k = x_k) \\ &= \Pr(\sum_{k=r+1}^{n} X_k \geq d-\sum_{j=1}^{r} x_j) \\ &= \Pr(S_{r+1} \geq d-z_r) \end{aligned} \tag{2.23}$$

By inserting (2.22) and (2.23) into (2.7) we get the following sampling probabilities for the k-out-of-n method:

$$\begin{aligned}
&\Pr(X_i = x_i \mid \phi_U(\boldsymbol{X}) - \phi_L(\boldsymbol{X}) = 1, \bigcap_{k=1}^{i-1} X_k = x_k) \qquad (2.24)\\
&= \frac{\Pr(S_{i+1} \geq d - z_i) - \Pr(S_{i+1} \geq n - c + 1 - z_i)}{\Pr(S_i \geq d - z_{i-1}) - \Pr(S_i \geq n - c + 1 - z_{i-1})} p_i^{x_i}(1-p_i)^{1-x_i}
\end{aligned}$$

The tail probabilities of the partial sums, $S_1, \ldots, S_n$, needed in (2.24) can be calculated from the distributions of $S_1, \ldots, S_n$ in $O(n^2)$ time. Assuming that this is done before running the simulations, each iteration of the simulation can be done in $O(n)$ time.

2.3 The factoring method

In this subsection we will use $s-p$-structures (series-parallel structures) as upper and lower bounds on system. An $s-p$-structure is a system which can be reduced to a single component by using $s-p$-reductions (series-parallel reductions). Thus, the reliability of such a structure can be calculated in $O(n)$ time. Since a disjoint path (or cut) sets structure can be reduced to a single component by using $s-p$-reductions, it follows that such systems are special cases of $s-p$-structures. Hence, the approach used in this subsection can be viewed as a generalization of the disjoint path and cut sets method. However, the method we use to construct the upper and lower bounds is different. We start out by introducing the following notation: Let $\boldsymbol{x}$ be a vector with index set $E = \{1, \ldots, n\}$, and let $A \subseteq E$. Then $(\mathbf{1}^A, \boldsymbol{x})$ denotes a vector obtained from $\boldsymbol{x}$ by replacing x_i by 1 for all $i \in A$. Similarly $(\mathbf{0}^A, \boldsymbol{x})$ denotes a vector obtained from $\boldsymbol{x}$ by replacing x_i by 0 for all $i \in A$. Furthermore, we denote by $\boldsymbol{x}^A$ the subvector obtained from $\boldsymbol{x}$ by deleting all entries x_i such that $i \notin A$.

Since every structure function is assumed to be nondecreasing, it follows that we always have:

$$\phi(\mathbf{0}^A, \boldsymbol{x}) \leq \phi(\boldsymbol{x}) \leq \phi(\mathbf{1}^B, \boldsymbol{x}), \qquad \forall A, B \subseteq E. \tag{2.25}$$

Thus, $\phi_L(\boldsymbol{x}^{E \setminus A}) = \phi(\mathbf{0}^A, \boldsymbol{x})$ is a lower bound on ϕ while $\phi_U(\boldsymbol{x}^{E \setminus B}) = \phi(\mathbf{1}^B, \boldsymbol{x})$ is an upper bound. Moreover, by choosing the sets A and B in a clever way, the reliabilities of the lower and upper bounds may be easy to compute. In the following we will assume that these sets are chosen so that the resulting lower and upper bounds are $s-p$-structures. It is easy to see that it is always possible to find such sets. From (2.5) we see that the best results are obtained when the sets A and B are chosen so that the difference $h(\mathbf{1}^B, \boldsymbol{X}) - h(\mathbf{0}^A, \boldsymbol{X})$ is as small as possible. As one might expect, however, finding the optimal sets can be very difficult. A simpler, but still reasonable approach is to try minimizing the cardinalities of A and B. In order to do so, a variant of the well-known *factoring algorithm* can be used. (See Huseby[8] or Satyanarayana and Chang[14].)

Assuming that the system under consideration is *regular* in the sense of Huseby[9], the computational complexity of the factoring algorithm is characterized by an invariant called the *domination.* The domination of a system with structure function ϕ, denoted by $D(\phi)$, can be defined as the absolute value of the coefficient of the highest order term in ϕ. The following are well-known properties of the domination of regular systems. (See Huseby[8] and Huseby[9].)

$$D(\phi) > 0, \quad \text{if and only if } \phi \text{ is a coherent structure.} \tag{2.26}$$

$$D(\phi) = 1, \quad \text{if and only if } \phi \text{ is a coherent } s-p\text{-structure.} \tag{2.27}$$

$$D(\phi) = D(\phi^r), \quad \text{if } \phi^r \text{ is an } s-p\text{-reduction of } \phi. \tag{2.28}$$

$$D(\phi) = D(\phi_{+e}) + D(\phi_{-e}), \quad \text{if } e \text{ is relevant,} \tag{2.29}$$

where $\phi_{+e} = \phi_{+e}(\boldsymbol{x}^{E\setminus e}) = \phi(1^e, \boldsymbol{x})$ and $\phi_{-e} = \phi_{-e}(\boldsymbol{x}^{E\setminus e}) = \phi(0^e, \boldsymbol{x})$.

We now explain heuristically how this can be applied to choose the set A used in the lower bound. Choosing B can be found by a dual approach. We start out with the original structure function ϕ and component set E, and assume that this system is coherent and not an $s-p$-structure. We then have to choose the first component, say $e \in E$, which will be left out of the system in order to produce a lower bound, denoted $\phi_{-e} = \phi_{-e}(\boldsymbol{x}^{E\setminus e}) = \phi(0^e, \boldsymbol{x})$. In order to minimize the number of components being left out, there are two things we need to take into account. If e is chosen so that ϕ_{-e} is not coherent, this is essentially equivalent to deleting e as well as the resulting irrelevant components from the system. Thus, whenever possible, we should choose e so that ϕ_{-e} is coherent. On the other hand we want to choose e so that ϕ_{-e} is as simple as possible, i.e., such that $D(\phi_{-e})$ is as small as possible. Since $D(\phi_{-e}) = D(\phi) - D(\phi_{+e})$, we should also choose e so that $D(\phi_{+e}) > 0$. That is, e should be chosen so that ϕ_{+e} is coherent as well. If ϕ does not contain any components in series or parallel, i.e., ϕ is $s-p$-complex, it can be shown that it is always possible to find e so that both ϕ_{-e} and ϕ_{+e} are coherent. (See Huseby[8].) If on the other hand ϕ is $s-p$-reducible, it may happen that no such single component exists. In such cases we must remove a minimal $s-p$-module of components instead (i.e., a minimal module of the system whose structure function is an $s-p$-structure). By repeating this process of removing either single components or $s-p$-modules, each time ensuring that the domination is reduced to a smaller but positive number, we finally arrive at a system which is a coherent $s-p$-structure which can be used as a lower bound for ϕ.

Since the factoring method allows to use general $s-p$-structures as upper and lower bounds instead of just disjoint path and cut sets structures, this method typically will produce better bounds than the disjoint path and cut sets method. However, with general $s-p$-structures we may not be able to carry out the sampling probability calculations as efficient as was the case with the disjoint path and cut sets method. To analyze this closer we represent the calculations needed to compute the reliability of an $s-p$-structure in a tree-structure (similar to a fault tree representation). The leaves of the tree represent the component reliabilities, while the other nodes of the tree represent reliabilities of $s-p$-modules of the system. To each of these nodes we assign either a product operator or a coproduct operator. The root of the tree represents the reliability of the system. We illustrate this by considering the $s-p$-structure shown in Figure 3. The corresponding tree-structure representing the calculations is shown in Figure 4.

Before sampling from the conditional distribution (2.7), the leaves of the tree are initialized with the component reliabilities, and the rest of the tree is calculated using the assigned operators. As we sample the component state variables, we update the tree by replacing the component reliabilities by the corresponding component state variables, and recalculate the nodes above. The number of operations needed for each update, is proportional to the length of the path from the updated leaf up to the root. For a perfectly balanced binary tree, the height is of order $\log(n)$. Thus for such systems, the simulations can be done in $O(n \log(n))$ time. On the other hand, for the disjoint path or cut sets systems considered earlier, all the paths are of length 2, so for this case we see that the simulations can be done in just $O(n)$ time as stated before. More generally, if the average path length is λ, then the simulations can be done in $O(n\lambda)$ time. In the worst case λ is proportional to n, in which case the order of the simulation procedure is $O(n^2)$.

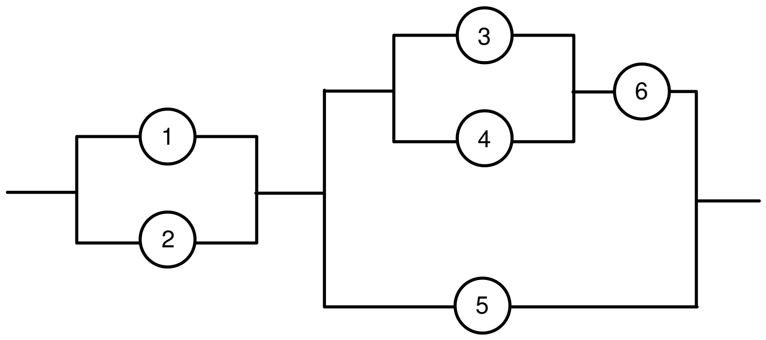

Figure 3: An $s - p$-structure of six components.

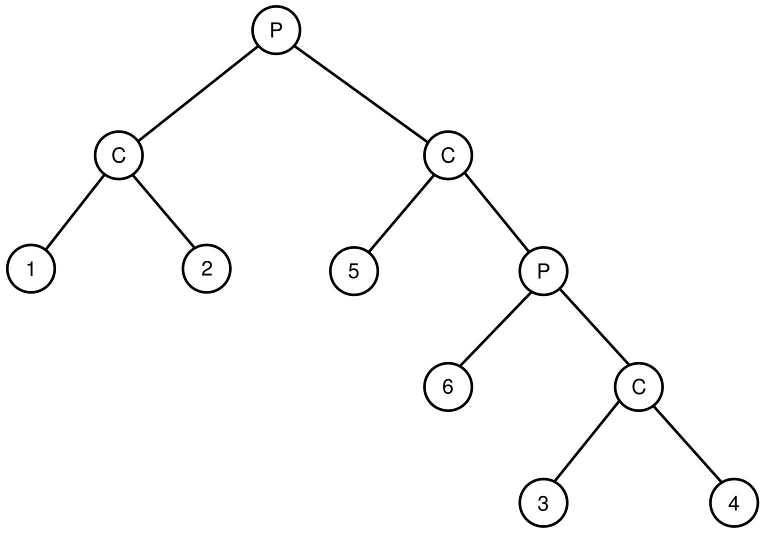

Figure 4: The calculations represented as a tree, where `P` represents a product operation while `C` represents a coproduct operation (i.e., the *ip*-operation).

Having presented three different methods for constructing upper and lower bounds on a system, a natural question is which method is the best. As one could expect, the answer to this question depends on the system under consideration. For a detailed study of this we refer to Vårli[15]. We close this section, however, by presenting a simple example where we compare the disjoint path and cut sets method to the factoring method. The system we consider is a 2-terminal undirected network system illustrated in Figure 5. The components of the system are the edges. The system is said to be functioning if the two terminal nodes can communicate through the network.

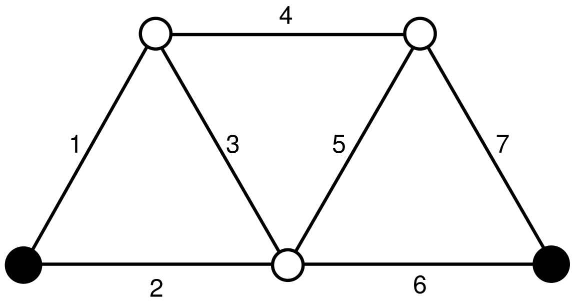

Figure 5: A 2-terminal undirected network system, where the terminals are the two black nodes.

Using the disjoint path and cut sets method we find the following collections of disjoint minimal path sets: $P_1 = \{1, 4, 7\}, P_2 = \{2, 6\}$, and disjoint minimal cut sets: $C_1 = \{1, 2\}, C_2 = \{6, 7\}$. (There are other possible choices as well, but these collections are optimal in the sense that they give us the best bounds.) Denoting the corresponding reliabilities of the lower and upper bounds by $h_{1,L}$ and $h_{1,U}$ respectively, we get that:

$$\begin{aligned} h_{1,L}(\boldsymbol{p}) &= (p_1 p_4 p_7) \amalg (p_2 p_6), \\ h_{1,U}(\boldsymbol{p}) &= (p_1 \amalg p_2)(p_6 \amalg p_7). \end{aligned} \tag{2.30}$$

On the other hand if we use the factoring method, we get a lower bound $s-p$-structure by replacing the reliability of component 4 by 0, and an upper bound $s-p$-structure by replacing the reliability of component 3 by 1. Denoting the lower and upper bounds by $h_{2,L}$ and $h_{2,U}$ respectively, we get that:

$$\begin{aligned} h_{2,L}(\boldsymbol{p}) &= [(p_1 p_3) \amalg p_2)][(p_5 p_7) \amalg p_6)], \\ h_{2,U}(\boldsymbol{p}) &= (p_1 \amalg p_2)(p_6 \amalg (p_7(p_4 \amalg p_5))). \end{aligned} \tag{2.31}$$

In Figure 6 we have plotted the four functions together with the exact reliability as functions of a common component reliability, p, which is varied from 0.0 to 1.0. We see that we do get better bounds by using the factoring method. On the other hand the upper and lower bounds of this method become somewhat more difficult to compute. The average computational path lengths for $h_{2,L}$ and $h_{2,U}$ are 16/6 and 17/6 respectively, compared to just 2 for $h_{1,L}$ and $h_{1,U}$. Still spending a little more time on each simulation is definitely worthwhile in this case thanks to the increased precision.

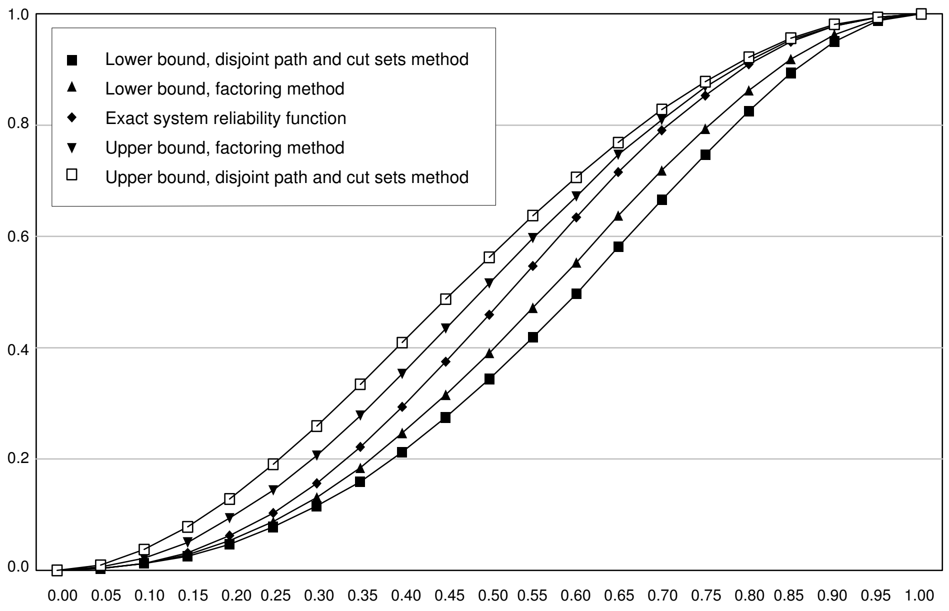

Figure 6: Reliability of upper and lower bound structures using disjoint path and cut sets method and factoring method.

3 Conditioning on the sum of the component state variables

For the remaining part of the paper we focus on the case where S is the sum of all the component state variables. This approach was taken in in Naustdal[12]. In this situation the random variable S takes values in the set $\{0, 1, \ldots, n\}$. Thus, the uncertain quantities we need to estimate, are:

$$\theta_s = E[\phi \mid S = s], \qquad s = 0, 1, \ldots, n \,. \tag{3.1}$$

As in the previous section, we need to have an efficient way of sampling from the conditional distribution of $\boldsymbol{X}$ given $S = s$, $s = 0, 1, \ldots, n$. In relation to this we need the use of the partial sums, $S_1, \ldots, S_n$, introduced in (2.20). Using these quantities, we can develop the algorithm for sampling from the distribution of $\boldsymbol{X}$ given $S = s$. The idea is simply to start out by sampling X_1 from the conditional distribution of $X_1 \mid S = s$. We then continue by sampling X_2 from $X_2 \mid S = s$, $X_1 = x_1$, where x_1 denotes the sampled outcome of X_1, and so on. This turns out to be easy noting that:

$$
\begin{aligned}
&\Pr(X_m = x_m \mid X_1 = x_1, \ldots, X_{m-1} = x_{m-1}, S = s) \qquad (3.2)\\
&\quad = \frac{\Pr(X_m = x_m, S = s \mid X_1 = x_1, \ldots, X_{m-1} = x_{m-1})}{\Pr(S = s \mid X_1 = x_1, \ldots, X_{m-1} = x_{m-1})}\\
&\quad = \frac{\Pr(X_m = x_m, S_{m+1} = s - \sum_{j=1}^{m} x_j)}{\Pr(S_m = s - \sum_{j=1}^{m-1} x_j)}\\
&\quad = \frac{p_m^{x_m}(1-p_m)^{1-x_m}\Pr(S_{m+1} = s - \sum_{j=1}^{m} x_j)}{\Pr(S_m = s - \sum_{j=1}^{m-1} x_j)},
\end{aligned}
$$

Assuming that the distributions of $S_1, \ldots, S_n$ are all calculated before running the simulations, by using (2.21), we see that all the necessary conditional probabilities can be calculated when needed during the simulations without imposing additional computational complexity. Note that we only calculate those conditional probabilities we really need along the way during the simulations, not the entire set of all possible conditional probabilities corresponding to all possible combinations of values of the X_j's. Thus, in each simulation run we calculate n probabilities, one for each X_j. Moreover, each probability can be calculated using a fixed number of operations (independent of n). Hence, it follows that sampling from the conditional distribution of $\boldsymbol{X}$ given $S = s$, can be done in $O(n)$ time.

The CMC-algorithm can now be summarized as follows: Divide the N simulations into $(n+1)$ groups, one for each θ_s, where the s-th group has size N_s, $s = 0, 1, \ldots, n$. The data in the s-th group is sampled from the conditional distribution of $\boldsymbol{X}$ given that $S = s$, and is used to estimate the conditional expectation θ_s, $s = 0, 1, \ldots, n$. Denoting the data in the s-th group by $\{\boldsymbol{X}_r, s : r = 1, \ldots, N_s\}$, θ_s is estimated essentially by using (1.4), and combined into the CMC-estimate by using (1.5).

A remaining problem is of course how to choose the sample sizes for each of the $(n+1)$ groups. One possibility is to proceed as we did in Section 1 and let $N_s \approx N \cdot \Pr(S = s)$, $s = 0, 1, \ldots, n$. In this case, however, it is possible to improve the results slightly. As in the previous section we denote the size of the smallest path set by d, and the size of the smallest cut set by c. By examining the system, it is often easy to determine d and c. Given these two numbers it is easy to see that $\theta_s = 0$ for $s < d$, and $\theta_s = 1$ for $s > n - c$. Moreover, $\mathrm{Var}(\phi \mid S = s) = 0$ for $s < d$, or $s > n - c$. Hence, there is no point in spending simulations on estimating θ_s for $s < d$, or $s > n - c$, so we let $N_s = 0$ for $s < d$, or $s > n - c$. As a result, we have more simulations to spend on the remaining quantities.

An extreme situation occurs when the system is a k-out-of-n-system, i.e., when $\phi(\boldsymbol{X}) = I(S \geq k)$. For such systems $d = k$, and $c = (n - k + 1)$. In this situation *all* the θ_s's are known. Specifically, $\theta_s = 0$ for $s < k$ and $\theta_s = 1$ for $s \geq k$. Thus, the CMC-estimate is equal to the true value of the reliability, and can be calculated without doing any simulations at all. The reason for this is of course that in this case ϕ depends on $\boldsymbol{X}$ only through S.

For all other nontrivial systems, however, it is easy to see that we always have that $d \leq n - c$. In order to ensure that we get improved results, we assume that $\Pr(d \leq S \leq n-c) > 0$, and let:

$$N_s \approx N \cdot \Pr(S=s) / \Pr(d \leq S \leq n-c), \qquad s = d, \ldots, n-c\,. \tag{3.3}$$

Using a similar argument as we did in (1.6) we now get that:

$$\operatorname{Var}(\hat{h}_{CMC}) \leq \Pr(d \leq S \leq n-c) \operatorname{Var}(\hat{h}_{MC}) \tag{3.4}$$

Hence, we see that if d and $(n-c)$ are close, i.e., there are few unknown θ_s's, the variance is reduced considerably.

4 System Reliability when All the Component State Variables Have Identical Reliabilities

If all the components in the system have the same reliability, i.e., $p_1 = \cdots = p_n = p$, it is possible to improve things even further. In this case we note that S has a binomial distribution. Moreover, the conditional distribution of $\boldsymbol{X}$ given S is given by:

$$\Pr(\boldsymbol{X} = \boldsymbol{x} \mid S = s) = \frac{p^{\sum_{i=1}^n x_i}(1-p)^{n-\sum_{i=1}^n x_i}}{\binom{n}{s} p^s (1-p)^{n-s}} = \frac{1}{\binom{n}{s}}\,, \tag{4.1}$$

for all $\boldsymbol{x}$ such that $\sum_{i=1}^n x_i = s$.
From this it follows that:

$$\theta_s = E[\phi \mid S = s] = \sum_{\{\boldsymbol{x} \mid \sum_{i=1}^n x_i = s\}} \phi(\boldsymbol{x}) \Pr(\boldsymbol{X} = \boldsymbol{x} \mid S = s) = \frac{b_s}{\binom{n}{s}}\,, \tag{4.2}$$

where $b_s =$ the number of path sets with s components, $s = 0, \ldots, n$.
Finally, the system reliability, h, expressed as a function of p, is given by:

$$\begin{aligned} h(p) &= \sum_{s=0}^n \theta_s \Pr(S = s) \\ &= \sum_{s=0}^n \frac{b_s}{\binom{n}{s}} \binom{n}{s} p^s (1-p)^{n-s} = \sum_{s=0}^n b_s p^s (1-p)^{n-s}. \end{aligned} \tag{4.3}$$

Note that the desired quantities, $\theta_0, \ldots, \theta_n$, do not depend on p. Thus, by estimating these quantities, we get an estimate of the entire $h(p)$-function. Moreover, we see that θ_s can be interpreted as the fraction of path sets of size s among all sets of size s, for $s = 0, 1, \ldots, n$. Thus, θ_s can be estimated by sampling random sets of size s and calculating the frequency of path sets among the sampled sets. It turns out that this can be done very efficiently as follows:

The idea is to sample sequences of sets of increasing size by sampling from the component set, $C = \{1, \ldots, n\}$, without replacements. The only set of size 0 is of course $\emptyset$. If $\emptyset$ is a path set, we have a trivial system which is always functioning. Obviously, $\theta_0 = 1$ in this case. Moreover, the reliability of such a system is 1. If $\emptyset$ is not a path set, it follows that $\theta_0 = 0$. In both cases we do not need to sample anything to determine the value of θ_0. Thus, we can focus on estimating $\theta_1, \ldots, \theta_n$ by sampling sets of size s, for $s = 1, \ldots, n$.

Let $\mathbf{T} = (T_1, \ldots, T_n)$ be a vector for storing results from the simulations. Before we start the simulations, this vector is initialized as $(0, \ldots, 0)$. The simulation algorithm now runs as follows:

For $i = 1, \dots, N$ **do**

Step 1 Sample a component from the set C, say component i_1, and define $A_1 = \{i_1\}$. If A_1 is a path set, T_1 is incremented with 1.

Step 2 Sample a component from the set $C \setminus A_1$, say component i_2, and define $A_2 = \{i_1, i_2\}$. If A_2 is a path set, T_2 is incremented with 1.

Step n Sample the last remaining component, say component i_n, and define $A_n = C$. If A_n is a path set, T_n is incremented with 1.

When all the simulations are carried out, the vector T contains the number of observed path sets of sizes $1, \dots, n$. From this we get the resulting estimates of $\theta_1, \dots, \theta_n$ simply as:

$$\theta_s = T_s/N, s = 1, \dots, n. \tag{4.4}$$

It is easy to see that sampling components randomly without replacements is equivalent to sampling the components according to a random permutation, $(i_1, \dots, i_n)$, of the component set $\{1, \dots, n\}$. Such a permutation can easily be generated using a well-known algorithm described in Knuth[10]. This algorithm works as follows:

Assume that we start out with a vector representing an initial arbitrary permutation of the components, say $(i_1, \dots, i_n)$. One may e.g., simply use $(1, \dots, n)$. We then generate a random permutation vector by modifying this initial permutation vector by running through n steps. In the k-th step we consider the element currently being the k-th element of the vector. We then sample a random index, j, in the interval $k, \dots, n$, and let the k-th and the j-th elements switch places in the permutation vector. When all the n steps are completed, it is easy to see that the resulting vector is indeed a random permutation (regardless of the initial state of the vector).

By using this algorithm, we are able to generate a complete sequence of n sets in just $O(n)$ time. However, since all sets in a sequence are derived from the same permutation, the θ_s-estimates become correlated. Still, the effect of this is more than compensated for since each θ_s-estimate now can be based on all N simulations, compared to just a subset of size N_s as was the case in the previous section.

When running this algorithm, much of the time is spent on the steps where the system state is evaluated. Thus, in order to minimize the running time of the algorithm, one would like to reduce the number of such evaluations. Since $A_1 \subset A_2 \subset \cdots \subset A_n$, it follows by the monotonicity of the structure function, ϕ, that if A_k is a path set, then so is A_{k+1}. Thus, we can stop evaluating the state of the system as soon as we have generated a path set, since we then know that all the remaining sets will be path sets as well. Still it is obviously important to carry out the system state evaluations as efficient as possible. The methodology for doing this may typically depend strongly on the representation of the given system. In order to show how this can be done, we consider two different classes of systems.

The first class of systems is called *threshold systems*, which have the property that the structure function can be written in the following form:

$$\phi(\boldsymbol{X}) = I\Big(\sum_{i=1}^{n} a_i X_i \geq b\Big), \tag{4.5}$$

where $a_1, \ldots, a_n$ and b are positive numbers. The a_i's are called the *weights* associated with the components, while the number b is called the *threshold value.* Notice that if $a_1 = \cdots = a_n = 1$ and $b = k$, we get a standard k-out-of-n-system. For such systems it is very easy to carry out all the system state evaluations. To do so we keep track of the sum of the weights associated with the sampled components. When a new component is sampled, the sum is updated by adding the weight associated with this component. When the sum of weights is greater than or equal to the threshold value, we know that we have generated a path set. Using this method, each system evaluation is carried out in constant time. Thus, in this case the total computational complexity of the simulation algorithm is just $O(nN)$, where n is the number of components in the system, and N is the number of simulations.

The second class of systems is called K-terminal undirected network systems. The components of such a system are the edges of an undirected network. An edge is functioning if its *end-nodes* can "communicate" through it. The system is functioning if a subset of the nodes (with K elements), called the *terminals*, can communicate through the network.

If all the edges are functioning, we assume that the network is connected, i.e., all nodes can communicate with each other. If only a subset of the edges are functioning, however, the node set is partitioned into a set of equivalence classes such that a pair of nodes can communicate if and only if they belong to the same equivalence class. Moreover, the system is functioning if and only if all the terminals belong to the same equivalence class.

In order to minimize the running time spent on system state evaluation, we maintain lists of nodes belonging to each equivalence class as we sample the edges. For each list we also keep track of the number of terminals contained in the list. When a new edge is sampled, we investigate its end-nodes. If they belong to the same equivalence class, the system state is not changed. On the other hand, if the end-nodes belong to different equivalence classes, we merge the two classes and calculate the number of terminals included in the merged class. When we arrive at an equivalence class containing K terminals, (i.e., all the terminals), we know that we have generated a path set.

The most time-consuming part of this algorithm is the merging of equivalence classes. Assuming that the classes are stored as linked lists, the actual merging of the lists can be done in constant time. However, in order to minimize the time spent on checking whether or not two nodes belong to the same class, the nodes should be marked with a reference to its class list. When two classes, say A and B, are merged, all the nodes in the smallest class, say B must be marked with a reference to the class list of A instead. Updating such references can be done in $O(v)$ time, where v is the number of nodes in the network. It should be noted, however, that this is a very pessimistic estimate, as in most cases the number of updating operations is much smaller. Anyway, the total computational complexity of the simulation algorithm is $O(nvN)$ in this case.

We close this section by illustrating the effect of the improved method on two specific examples. The system considered in the first example, is a threshold system with 20 components with the following structure function:

$$\begin{aligned}\phi(\boldsymbol{X}) = I(&2X_1 + 2X_2 + 3X_3 + 3X_4 + 3X_5 + 3X_6 + 4X_7 + 4X_8 \\ &+4X_9 + 4X_{10} + 4X_{11} + 4X_{12} + 4X_{13} + 5X_{14} + 5X_{15} \\ &+5X_{16} + 5X_{17} + 5X_{18} + 5X_{19} + 6X_{20} \geq 49) \qquad (4.6)\end{aligned}$$

The resulting estimates using the simple Monte Carlo method are shown in Figure 7, while the corresponding results for the improved CMC-method are shown in Figure 8.

For this type of system we see that the improved method actually produces nearly perfect results already after 10 simulations, while the corresponding results for the simple Monte Carlo method, is not satisfactory even after 100 simulations. In addition the simple Monte Carlo method requires separate estimates for each value of p. Thus, in order to obtain estimates for say 99 different values of p, this method actually uses 9900 simulations with significantly poorer results than we get with only 10 simulations using the improved CMC-method.

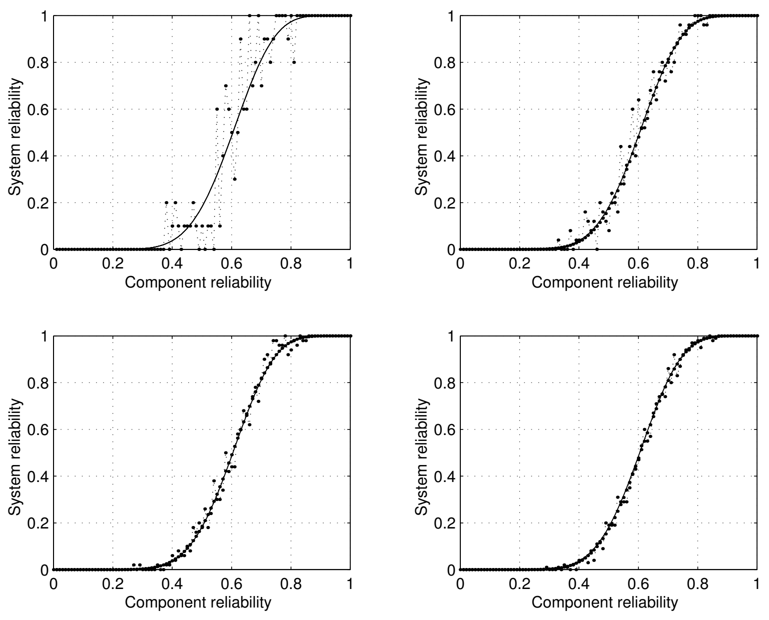

Figure 7: System reliability of a threshold system as a function of component reliability (dotted line), estimated using a simple Monte Carlo method with 10 (upper left), 25 (upper right), 50 (lower left) and 100 (lower right) simulations. The true reliability function is shown as a solid line.

In the second example we consider a simple bridge structure with 5 components shown in Figure 9.

The components of the system are the edges of the network, denoted by $1, 2, \ldots, 5$. The system is functioning if the terminal nodes, S and T, can communicate through the network.

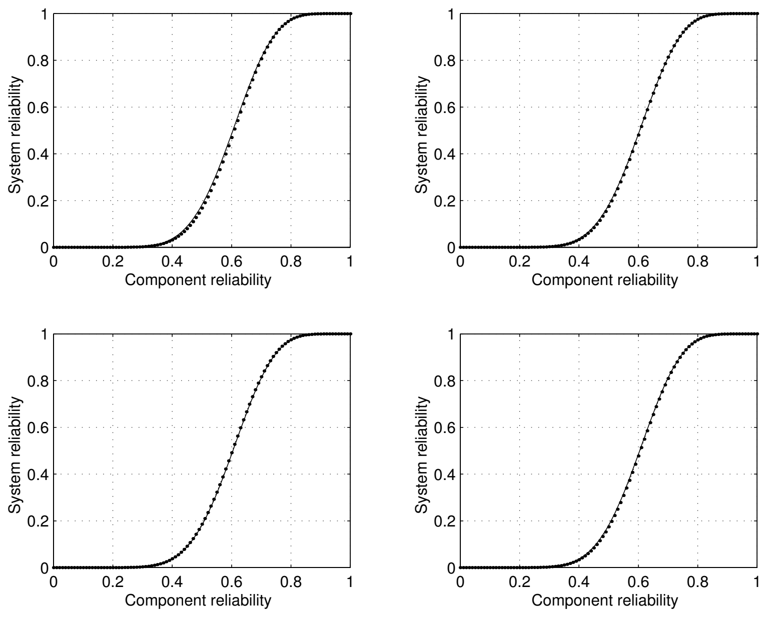

Figure 8: System reliability of a threshold system as a function of component reliability (dotted line), estimated using the improved CMC-method with 10 (upper left), 25 (upper right), 50 (lower left) and 100 (lower right) simulations. The true reliability function is shown as a solid line.

Our objective is to estimate the reliability h as a function of the common component reliability p. To illustrate the strength of the CMC-estimate, we will compare it to a simple Monte Carlo estimate. When using the latter method, one has to estimate h separately for each value of p.

In Figure 10 we have plotted the estimated values of h for $p = 0.01, \dots, 0.99$ obtained after 10, 25, 50 and 100 simulations for each p-value.

By using the improved method, we only need to estimate the θ_s's. Based on these estimates the entire curve can be calculated. The corresponding results are shown in Figure 11.

While the simple Monte Carlo method produces a jagged curve even after 100 simulations, the improved CMC-method gives excellent results already after 50 runs. In fact the improved method always produces a smooth S-shaped curve which rapidly converges to the true curve.

5 General Sequential Sampling Methods

The sampling method presented in the previous section was an example of what we shall refer to as a *sequential sampling method*. The strength of this method makes it tempting to investigate whether or not this idea can be generalized to a situation where the components have different reliabilities. A generalized sequential sampling method can be described as follows:

Let $C = \{1, \dots, n\}$ be a set of components. We then sample an ordered

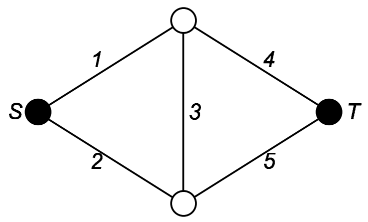

Figure 9: A bridge structure.

sequence, $(i_1, \ldots, i_n)$ of components without replacements as follows: Assume that we have sampled the s first components in the sequence, i.e., $i_1, \ldots, i_s$, and denote the corresponding unordered set $\{i_1, \ldots, i_s\}$ by A_s. Then the next component is sampled from the set of remaining components, i.e., $C \setminus A_s$, with probability:

$$\alpha_{A_s,i} = \Pr\,(\text{The } (s+1)\text{-th sampled component is } i \mid A_s), \quad \text{for all } i \in C \setminus A_s. \tag{5.1}$$

The probability of a sampling a given ordered sequence, $(i_1, \ldots, i_n)$ is then:

$$\alpha_{A_0,i_1} \cdot \alpha_{A_1,i_2} \cdots \alpha_{A_{n-1},i_n}, \tag{5.2}$$

where $A_0 = \emptyset$.

The sequential sampling method is characterized by the sampling probability distributions used in each sampling step, i.e., by the $\alpha_{A_s,i}$'s.

Using such a sampling method, the probability that after having sampled s components, we have sampled the set A, where $|A| = s$, is given by:

$$\Pr(A_s = A) = \sum_{(i_1,\ldots,i_s)\in\pi(A)} \alpha_{\emptyset,i_1} \cdots \alpha_{\{i_1\},i_2} \cdots \alpha_{\{i_1,\ldots,i_{s-1}\},i_s}\ , \tag{5.3}$$

where $\pi(A)$ denotes the set of all permutations of A.

Now, ideally we want the sequential sampling method to produce a sequence of sets $A_1, \ldots, A_n$ with the "correct" probabilities. That is, we want the $\alpha_{A_s,i}$'s to be chosen such that:

$$\Pr(A_s = A) = \Pr(\boldsymbol{X} = \boldsymbol{x}(A) \mid S = s), \qquad s = 1, \ldots, n \tag{5.4}$$

where $\boldsymbol{x}(A)$ denotes the vector $\boldsymbol{x} = (x_1, \ldots, x_n)$ such that $x_i = 1$ if $i \in A$, and 0 otherwise. If such a sampling method can be found, we could use the same method as in the previous section in order to obtain estimates for $\theta_0, \ldots, \theta_n$.

In the special case considered in the previous section where all the components have equal reliability, this is easily accomplished by using a simple *uniform sampling method*, i.e., by letting:

$$\alpha_{A_s,i} = \frac{1}{n - |A_s|} = \frac{1}{n - s}\ , \qquad \text{for all } i \in C \setminus A_s \tag{5.5}$$

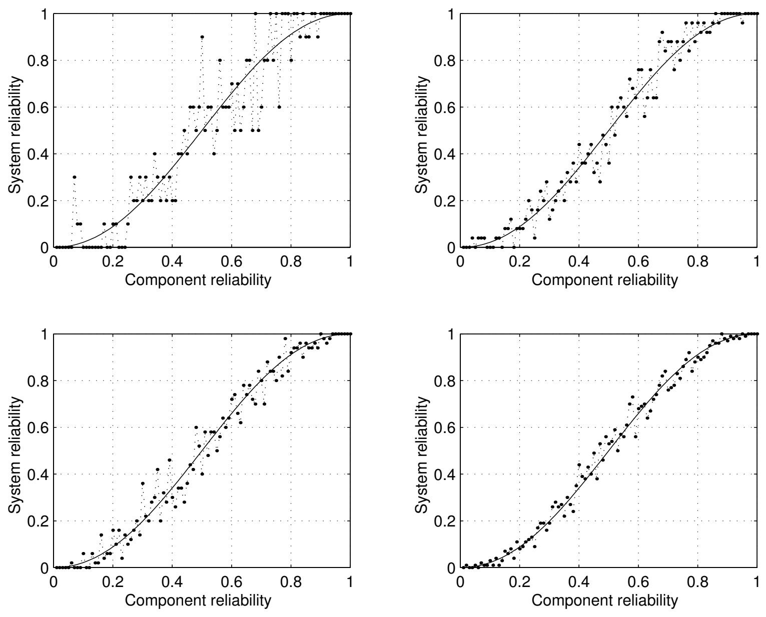

Figure 10: System reliability of a bridge system as a function of component reliability (dotted line), estimated using a simple Monte Carlo method with 10 (upper left), 25 (upper right), 50 (lower left) and 100 (lower right) simulations. The true reliability function is shown as a solid line.

With this choice we get from (5.3) by applying (4.1) that:

$$\begin{aligned}\Pr(A_s = A) &= \frac{s!}{n(n-1)\cdots(n-s+1)} = \frac{1}{\binom{n}{s}} \\ &= \Pr(\boldsymbol{X} = \boldsymbol{x}(A) \mid S = s), \qquad s = 1, \ldots, n\ .\end{aligned} \tag{5.6}$$

However, when the components have different reliabilities, it is much more difficult to find the right sampling probability distributions. In principle it is possible to establish a set of equations for the $\alpha_{A_s,i}$'s from which these quantities could be derived. Still the very large number of unknowns makes this approach unfeasible. In fact it is not even clear whether it is possible in general to find any solution to the equations.

To proceed we will instead assume that we have found a sequential sampling method which is a reasonable approximation to the correct distribution, and then use importance sampling to correct the results. More specifically we introduce the true conditional distribution function defined for all vectors $\boldsymbol{x}$ of binary variables and for $s = 0, 1, \ldots, n$:

$$f_s(\boldsymbol{x}) = \Pr(\boldsymbol{X} = \boldsymbol{x} \mid S = s). \tag{5.7}$$

Moreover, we introduce the corresponding sequential sampling distribution

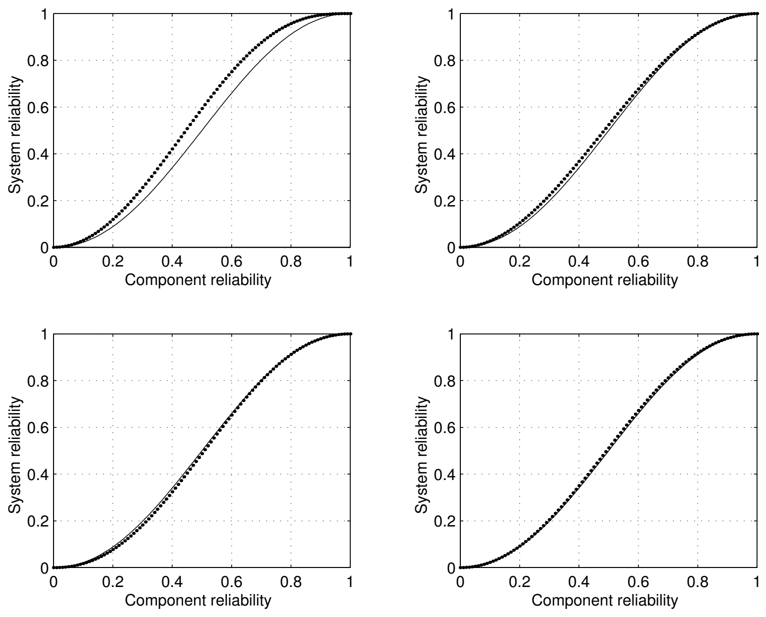

Figure 11: System reliability of a bridge system as a function of component reliability (dotted line), estimated using the improved CMC-method with 10 (upper left), 25 (upper right), 50 (lower left) and 100 (lower right) simulations. The true reliability function is shown as a solid line.

function also defined for all vectors $\boldsymbol{x}$ of binary variables and for $s = 0, 1, \ldots, n$:

$$\begin{aligned} g_s(\boldsymbol{x}) &= \Pr(A_s = A(\boldsymbol{x})) \\ &= \sum_{(i_1,\ldots,i_s)\in\pi(A(\boldsymbol{x}))} \alpha_{\emptyset,i_1} \cdot \alpha_{\{i_1\},i_2} \cdots \alpha_{\{i_1,\ldots,i_{s-1}\},i_s} , \end{aligned} \tag{5.8}$$

where $A(\boldsymbol{x}) = \{i : x_i = 1\}$.

We assume that $g_s(\boldsymbol{x})$ is a "close" approximation to $f_s(\boldsymbol{x})$, and that $g_s(\boldsymbol{x}) \neq 0$ for all $\boldsymbol{x}$. Under this assumption an unbiased estimator for the unknown quantity θ_s is given by the following importance sampling estimator:

$$\hat{\theta}_{s,I} = \frac{1}{N} \sum_{i=1}^{N} \phi(\boldsymbol{x}_{i,s}) \frac{f_s(\boldsymbol{x}_{i,s})}{g_s(\boldsymbol{x}_{i,s})} , \qquad s = 1, \ldots, n \tag{5.9}$$

where $\boldsymbol{x}_{1,s}, \ldots, \boldsymbol{x}_{N,s}$ are generated from the distribution g_s.

A serious difficulty with this method is that calculating $g_s(\boldsymbol{x})$ implies computing a sum of $s!$ terms. If s is large, this takes a very long time, and as a result the efficiency of the algorithm is destroyed. In order to avoid this problem, we will instead extend our stochastic experiments by considering ordered samples.

An ordered set, $A^o = (i_1, \ldots, i_s)$, sampled according to the sequential method, has the following probability of occurring:

$$\Pr(A^o = (i_1, \ldots, i_s)) = \alpha_{A_0,i_1} \cdot \alpha_{A_1,i_2} \cdots \alpha_{A_{s-1},i_s} , \tag{5.10}$$

where $A_0 = \emptyset$, $A_1 = \{i_1\}$, $\ldots$, $A_{s-1} = \{i_1, \ldots, i_{s-1}\}$. We denote this probability by $g_s(A^o)$.

It is easy to see that the matching "correct" probability, which we denote by $f_s(A^o)$, is given by:

$$f_s(A^o) = (s!)^{-1} \cdot f_s(\boldsymbol{x}(A^o)), \tag{5.11}$$

where $\boldsymbol{x}(A^o)$ is defined in the same way as for unordered sets.

An unbiased importance sampling estimator for θ_s is then given by:

$$\hat{\theta}_{s,I} = \frac{1}{N} \sum_{i=1}^{N} \phi(\boldsymbol{x}(A^o_{i,s})) \frac{f_s(A^o_{i,s})}{g_s(A^o_{i,s})}\ , \qquad s = 1, \ldots, n, \tag{5.12}$$

where $A^o_{1,s}, \ldots, A^o_{N,s}$ are generated from the distribution $g_s(A^o)$.

The variance of this estimator is:

$$\mathrm{Var}(\hat{\theta}_{s,I}) = \frac{1}{N} \mathrm{Var}\left(\phi(A^o_s)\frac{f_s(A^o_s)}{g_s(A^o_s)}\right), \tag{5.13}$$

where A^o_s is distributed according to the sequential sampling distribution, g_s. From this it follows that this variance is bounded, and that it converges towards 0 as N goes to infinity. Using standard methods it can easily be shown that the estimator will converge almost surely to the correct value.

By choosing the sequential sampling distribution, g_s, in a clever way one may reduce the variance of the importance sampling estimator. Still finding the optimal distribution is typically a difficult task. This is especially true in this case, since we need a sampling distribution which produces stable results for all the unknown quantities simultaneously. A reasonable strategy, however, can be to look for distributions which are good approximations to f_s. In particular one would like the ratio between f_s and g_s to be as stable as possible. This ratio is referred to as the *importance function*, and can be interpreted as a correction factor compensating for the fact that we sample from an incorrect distribution.

In general defining a sequential sampling method involves specifying a very large number of sampling probability distributions, one for each possible sampling situation occurring during the sampling sequences. In order to simplify this process, we will restrict ourselves to considering classes of distributions which can be described with only a limited set of parameters. One such class is *weighted sampling without replacement.* This class is constructed by assigning weights to each component. Let w_i be the weight assigned to component i, $i = 1, \ldots, n$. The sampling probability distributions are then given by the following:

$$\alpha_{A_s,i} = \frac{w_i}{\sum_{j \notin A_s} w_j}\ , \qquad \text{for all } i \in C \setminus A_s \tag{5.14}$$

We observe that if all the component weights are equal, this method reduces to the uniform sampling method.

In order to apply this scheme, we need an efficient way of generating samples from such distributions. We recall that for the case of uniform sampling, we used the algorithm described in Knuth[10]. It is possible to generalize this algorithm to a situation where the components have unequal weights. The critical issue here is to make sure that at each step in the procedure, the sampling probabilities of each of the remaining components are proportional to their weights. This is easily done as follows:

Assume that the set of remaining components from which we want to sample from is $\{i_1, \dots, i_k\}$, and denote the sum of weights of these components by W_k. We then generate a random number from the continuous uniform distribution on the interval $[0, W_k]$, and choose the next sampled component, say component i_j, so that j is the smallest number satisfying:

$$\sum_{r=1}^{j} w_j \geqq W_k \tag{5.15}$$

It is easy to see that this procedure makes the sampling probabilities proportional to their weights as requested. The problem with this method, however, is that searching for the component i_j may take as much as $O(n)$ time. Since this has to be done for each sampled component, the computational complexity of the generalized algorithm becomes $O(n^2)$. This means that we can just as well use the method introduced in (3.2).

Fortunately, it is possible to find a better solution.

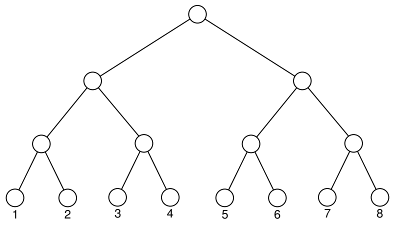

Figure 12: References to the components of a system stored as leaves in a binary tree.

Assume that references to all the components of the system are stored as leaves in a binary tree, as in Figure 12. For each leaf in the tree we also assign the weight of the corresponding component. We then proceed by assigning weights to the nodes in the level above the leaves by adding the weights of the attached leaves. The same procedure is applied to the next level of nodes, and so on all the way up to the root of the tree. Thus, the sum of all the weights assigned to the nodes in a level is the same for all levels. In particular, the weight assigned to the root is equal to the sum of the weights of all the leaves. As an example consider the binary tree shown in Figure 12. Assume that the weights assigned to the eight leaves in the tree, are $w_1, \dots, w_8$ respectively. Then the four nodes at the next level will be assigned the weights $w_1 + w_2$, $w_3 + w_4$, $w_5 + w_6$, $w_7 + w_8$, respectively. Similarly, the two nodes above these will be assigned the weights $w_1 + \cdots + w_4$, $w_5 + \cdots + w_8$, respectively. Finally, the root will be assigned the weight $w_1 + \cdots + w_8$.

If the number of components is a power of 2, the tree can be constructed to be perfectly balanced. If not, it may happen that some of the nodes only have a single node attached to it at the level below. Still it is always possible to structure the tree so that the total number of nodes in the tree is $O(n)$ and the height of the tree is $O(\log(n))$. Thus, in particular, all the weight assignments can be done in $O(n)$ time.

Having assigned all the weights we now sample the first component by generating a random walk from the root of the tree down towards a leaf following a branch of the tree. At each step of the walk there are (at most) two possible nodes to choose between. The choice is made at random so that the probability of each node is proportional to the weight assigned to it. It is easy to see that this implies that the probability of ending up at a particular leaf is proportional to its weight. Moreover, the number of steps in the walk is $O(\log(n))$.

Having sampled the first leaf and component, we repeat the same procedure for the second leaf and component. However, in order to avoid ending up at the same leaf, we replace its weight by zero, and update all the nodes above it accordingly. The number of updates needed is equal to the height of the tree, i.e., $O(\log(n))$.

By repeating the same process over and over until all the components are sampled, the procedure is completed. The computational complexity of this procedure is $O(n \log(n))$. Thus, this algorithm is indeed better than the first one.

We close this section by presenting an example of a weighted sampling without replacement scheme. The idea behind this approach is to use a sampling distribution where components with high reliabilities have a greater chance of being selected than those with low reliabilities. This is achieved by using the component reliabilities as weights. That is, we use the following distributions:

$$\alpha_{A_s,i} = \frac{p_i}{\sum_{j \notin A_s} p_j}, \qquad \text{for all } i \in C \setminus A_s \tag{5.16}$$

Now, assume that $A_s^o = (i_1, \ldots, i_s)$, and let:

$$A_0 = \emptyset, A_1 = \{i_1\}, A_1 = \{i_1, i_2\}, \ldots, A_{s-1} = \{i_1, \ldots, i_{s-1}\}. \tag{5.17}$$

Then g_s can be written as:

$$g_s(A_s^o) = \frac{\prod_{i \in A_s^o} p_i}{\prod_{k=0}^{s-1} \sum_{j \notin A_k} p_j} \tag{5.18}$$

After a few intermediate steps, we arrive at the following importance function:

$$\frac{f_s(A_s^o)}{g_s(A_s^o)} = \frac{\left(\prod_{k=0}^{s-1} \frac{1}{n-k} \sum_{j \notin A_k} p_j\right) \left(\prod_{i \notin \{A_s^o\}} (1 - p_i)\right)}{\binom{n}{s}^{-1} \Pr(S = s)} \tag{5.19}$$

We observe that the first factor of the numerator is a product of s average component reliabilities, where the averages are taken over the sets $(C \setminus A_0), (C \setminus A_1), \ldots, (C \setminus A_s)$. This averaging tends to make the importance function more stable. The second factor, however, may vary a lot. Assume e.g., that the system under consideration contains ten components of which five of them have reliabilities equal to 0.9, while the remaining components have reliabilities 0.99. If $s = 5$, we see that the second factor of the numerator varies between 10^{-5} and 10^{-10}. As a result the variance of the importance function becomes very large.

The above sampling method along with some others are studied in more detail in Naustdal[12]. The main result from this study is that the proposed methods work very well as long as the component reliabilities are not too different. However, when there are considerable differences between these numbers, the importance function becomes unstable, and the resulting estimates converge much slower.

6 Conclusions

In this paper we suggest several improved simulation methods with application to system reliability estimation. For the case where all the components have the same reliability, the best approach is to apply the sequential sampling method described in Section 4. If the component reliabilities are not too different, the proportional sequential sampling method described in Section 5 combined with importance sampling produces good results. When there are significant differences between these reliabilities, however, the unsequential conditional Monte Carlo method suggested in Sections 2 and 3 appears to be better. Still we believe that the importance sampling methods can be improved further by finding sampling methods which produce closer approximations to the "correct" distributions.

We close this section by briefly indicating how the methods of Section 2 and 3 can be combined. In order to do so we start out by using the factoring method to derive upper and lower bounds on the structure function. When we discussed this method in Section 2, we allowed the sets A and B to be different. Now, however, we assume that we have found a common set, say $D \subseteq E$, such that both $\phi_L(\boldsymbol{x}^{E \setminus D}) = \phi(\mathbf{0}^D, \boldsymbol{x})$ and $\phi_U(\boldsymbol{x}^{E \setminus D}) = \phi(\mathbf{1}^D, \boldsymbol{x})$ are $s-p$-structures. This is always possible. We may e.g., let $D = A \cup B$, where A and B are derived using the method explained in Section 2.

We then define $Z_D = \sum_{i \in D} X_i$, and consider the vector $S = (\phi_L, \phi_U, Z_D)$. We observe that if $Z_D = 0$, then we know that $\phi = \phi_L$, while if $Z_D = |D|$, it follows that $\phi = \phi_U$. Thus, the only values of S we need to condition on during the simulations are $\{S = (0, 1, z) : z = 1, \ldots, |D| - 1\}$. This approach combines the best properties from the upper and lower bound method as well as the sum method. Thus, one would expect the results to be even better than the ones obtained using the methods suggested in this paper. We will return to this approach in a future paper.

Acknowledgments

The authors are grateful to Professor Bo Lindqvist and Professor Bent Natvig for helpful discussions.

References

[1] R. E. Barlow and F. Proschan, *Statistical theory of reliability and life testing. Probability models* Holt, Rinehart and Winston, New York, (1975).

[2] G. Broström, and L. Nilsson, Acceptance/rejection sampling from the conditional distribution of independent discrete random variables, given their sum, *Statistics* **34**, 247 (2000).

[3] H. Cancela and M. El Khadari, A recursive variance-reduction algorithm for estimating communication-network reliability, *IEEE Transactions on Reliability* **R-44**, (1995).

[4] H. Cancela and M. El Khadari, Series-parallel reductions in Monte Carlo network reliability evaluation, *IEEE Transactions on Reliability* **R-47**, (1998).

[5] H. Cancela and M. El Khadari, The recursive variance-reduction simulation algorithm for network reliability evaluation, *IEEE Transactions on Reliability* **R-52**, (2003).

[6] G. S. Fishman, A Monte Carlo sampling plan for estimating network reliability, *Operations Research* **34**, (1986).

[7] G. S. Fishman, A comparison of four Monte Carlo methods for estimating the probability of $s - t$ connectedness, *IEEE Transactions on Reliability* **R-35** (1986).

[8] A. B. Huseby, A Unified Theory of Domination and Signed Domination with Application to Exact Reliability Computations, Statistical Research Report **3**, University of Oslo (1984).

[9] A. B. Huseby, Domination theory and the Crapo β-invariant, *Networks* **19** 135-149, (1989).

[10] D. E. Knuth, *The art of computer programming, Vol. 2, Seminumerical algorithms*, Addison-Wesley, Reading, Mass. second edition, (1981).

[11] B. H. Lindqvist and G. Taraldsen, *Monte Carlo Conditioning on a Sufficient Statistics*, Preprint Statistics **9**, Norwegian University of Science and Technology (2001).

[12] M. Naustdal, *Forbedrede betingede Monte Carlo metoder i pålitelighetsberegninger*, (in Norwegian) Cand. Scient. thesis, Department of Mathematics, University of Oslo. (2001).

[13] L. Nilsson, *On the simulation of conditional distributions in the Bernoulli case*, Research report **14**, Department of Mathematical Statistics, Umeå University, Sweden (1997).

[14] A. Satyanarayana and M. K. Chang, Network Reliability and the Factoring Theorem, *Networks* **13** 107-120, (1983).

[15] I. D. Vårli, *Forbedrede Monte Carlo metoder i pålitelighetsberegninger*, (in Norwegian) Cand. Scient. thesis, Department of Mathematics, University of Oslo (1996).