An adaptive subset simulation algorithm for system reliability analysis with discontinuous limit states

Jianpeng Chan, Iason Papaioannou, Daniel Straub

Engineering Risk Analysis Group, Technische Universität München, Arcisstr. 21, 80290 München, Germany

Abstract

Many system reliability problems involve performance functions with a discontinuous distribution. Such situations occur in both connectivity- and flow-based network reliability problems, due to binary or multi-state random variables entering the definition of the system performance or due to the discontinuous nature of the system model. When solving this kind of problems, the standard subset simulation algorithm with fixed intermediate conditional probability and fixed number of samples per level can lead to substantial errors, since the discontinuity of the output can result in an ambiguous definition of the sought percentile of the samples and, hence, of the intermediate domains. In this paper, we propose an adaptive subset simulation algorithm to determine the reliability of systems whose performance function is a discontinuous random variable. The proposed algorithm chooses the number of samples and the intermediate conditional probabilities adaptively. We discuss two MCMC algorithms for generation of the samples in the intermediate domains, the adaptive conditional sampling method and a novel independent Metropolis-Hastings algorithm that efficiently samples in discrete input spaces. The accuracy and efficiency of the proposed algorithm are demonstrated by a set of numerical examples.

Keywords: Subset simulation, System reliability analysis, Limit state function with discontinuous distribution, Conditional sampling, independent Metropolis-Hastings

1. Introduction

Infrastructure networks, such as power grids and water supply systems, deliver essential services to society. Failures of such networks can have severe consequences. Quantification of the probability of survival or, conversely, the probability of failure of such systems is essential in understanding and managing their reliability; this is the main purpose of network system reliability assessment.

For reliability analysis purposes, the performance of the system can be assessed by the limit state function (LSF), also known as performance function or structure function, $g(\boldsymbol{X})$. $\boldsymbol{X}$ is an n-dimensional vector of random variables with joint cumulative distribution function (CDF) $F_{\boldsymbol{X}}$ and represents the uncertainty in the model input. By convention, failure of the system occurs for all system states $\boldsymbol{x}$ for which $g(\boldsymbol{x}) \leqslant 0$. The probability of failure of the system is defined as

$$p_f \triangleq \mathbb{P}(g(\boldsymbol{X}) \leqslant 0) = \int_{g(\boldsymbol{x})\leqslant 0} \mathrm{d}F_{\boldsymbol{X}}(\boldsymbol{x}) \tag{1}$$

The vector of basic random variables $\boldsymbol{X}$ entering the definition of the LSF of network systems usually contains discrete random variables, which results in a LSF with discontinuous distribution. This is due to the fact that the performance of the network is often calculated through a function of a large number of binary or multi-state components. Moreover, real-world infrastructure networks are often designed to be highly reliable. This leads to high-dimensional reliability assessment problems with small failure probabilities [1].

Network performance is often measured through connectivity or 'travel time'(or flow) [2]. In connectivity-based problems, one evaluates the probability that a given set of nodes are connected, given that each component of the network fails with some probability. Typically, both the system performance and the component state are modeled as binary random variables. In this context, $g(\boldsymbol{X})$ is known as the structure function [3]. A set of sampling-based methods have been proposed for such kind of problems (e.g., [4, 5, 6, 7, 8, 9, 10, 11]), and a comparative study can be found in [12, 13].

In this paper, we focus on flow-based problems where the system performance and/or the component are typically modeled as multi-state or continuous random variables instead of binary ones, and, hence, most of the sampling techniques tailored for connectivity-based problems cannot be implemented

directly. One of the major concerns in this area is the maximum flow that a stochastic network can deliver, i.e., the probability that the maximum flow from one or more source nodes to one or more terminal nodes is less than a predefined demand level. A number of sampling based methods have been proposed for this type of problems [14, 15, 16, 17, 18, 19, 20, 21, 22]. However, all these methods assume that the edge capacities are independent and discrete random variables, which is often unrealistic. [1] employs the standard subset simulation (SuS) algorithm [23] to efficiently solve maximum-flow reliability problems, where both the edge capacity and the network performance are modeled as continuous random variables. [24] use the standard SuS algorithm in reliability analysis of gas pipelines. However, as discussed in this paper, the adaptive approach of the standard SuS for determining the intermediate levels is not suitable for LSFs with discontinuous distribution, which is the case for most network reliability problems. As an example, Fig. 1 shows the CDF of the LSF of the IEEE39 bus benchmark system (described in Section 5.4). The CDF is discontinuous with 'jumps'. To overcome this

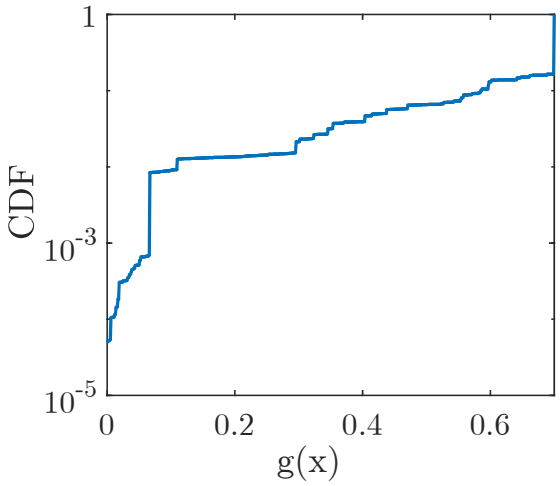


Figure 1: CDF of the LSF of the IEEE39 bus benchmark system of Section 5.4

limitation, one may construct a problem equivalent to the original one but with LSFs with continuous distribution, and then use SuS to solve this equivalent problem. This approach has been explored by Ching and Hsu [2] for connectivity-based problems, where a virtual random walk model is solved to get a continuous proxy of the original binary connectivity. Typically, such transformations need to be derived for the problem at hand.

The generalized splitting method [25] has also been employed to solve both connectivity- [10] and flow-based problems [26] in combination with tailored and efficient Gibbs samplers. It should be stressed that the determination of the intermediate levels in the generalized splitting method is through a pilot run of the adaptive multilevel splitting algorithm [25, 10, 27], which is essentially the standard SuS algorithm. Therefore, transformation of discontinuously distributed LSF to continuous one is also needed for these approaches. The basic idea of SuS is to express the probability of failure as a product of larger conditional probabilities of a set of intermediate nested events. Two ingredients of the SuS algorithm are essential for obtaining an accurate and efficient estimator. The first is the efficient simulation of conditional samples, which is achieved through Markov Chain Monte Carlo (MCMC) methods [23, 28]. The second is the proper choice of the intermediate events. Similarly to the cross entropy method [29, 30], the intermediate failure events in SuS are chosen adaptively, so that the estimates of the conditional probabilities equal a predefined value p_0. This is achieved through generating a fixed number of samples in each conditional level, sorting the samples according to their LSF values and determining the p_0-percentile of the samples, which is set as the threshold defining the next intermediate failure event. When solving network reliability problems, the discontinuous nature of the LSF can result in a large number of samples in a certain conditional level having the same LSF value. In such cases, the standard SuS method will result in an ambiguous definition of the intermediate domains. In extreme conditions, all samples generated in a certain level might have the same LSF value, in which case the sample process can get stuck and might not reach the failure domain.

To address this issue, we introduce a novel variant of SuS called *adaptive effort subset simulation* (aE-SuS) method. Our method chooses the number of samples per level and the respective conditional probability adaptively to ensure that an adequate number of samples fall in the subsequent intermediate domain. Compared with other non-sampling based methods (e.g., [31, 32, 33, 34, 35]), the proposed method facilities using advanced deterministic network analysis algorithms considering complex network dynamics like cascading failure. On the other hand, owing to its sampling nature, the aE-SuS algorithm may require a large number of simulations to achieve an acceptable result. It should be stressed that the proposed aE-SuS algorithm is applicable for dependent input random variables and any MCMC algorithm that enables efficient sampling of the intermediate conditional distributions

can be combined with the proposed algorithm.
The paper is organized as follows: Section 2 gives a brief introduction to the standard SuS. Section 3 discusses two MCMC algorithms in the context of network reliability assessment. Section 4 introduces the basic idea as well as the implementation details of the aE-SuS method. In Section 5, the performance of the proposed algorithm is illustrated by a set of numerical examples, a one-dimensional multi-state problem, a multidimensional flow-based problem with combined continuous and binary capacities, a binomial experiment with small success probability, and a benchmark power transmission network system. The paper closes with the conclusions in Section 6.

2. Standard subset simulation

2.1. Brief introduction of subset simulation

The basic idea of SuS is to express the rare failure event $F = \{\boldsymbol{x} : g(\boldsymbol{x}) \leqslant 0)\}$ as the intersection of a sequence of nested intermediate events $F_1 \supset F_2 \supset \cdots \supset F_m$. Owing to the nestedness of the intermediate events, the failure event can be expressed as $F = \cap_{l=1}^{m} F_l$. The failure probability can then be decomposed as the following product of conditional probabilities:

$$\mathbb{P}(F) = \prod_{l=1}^{m} \mathbb{P}(F_l|F_{l-1}) \tag{2}$$

where F_0 is the certain event. Ideally, the intermediate events are selected such that each conditional probability is large, typically $\geqslant 0.1$. In this way, the original problem of estimating a small probability is transformed to a sequence of m intermediate problems of evaluating larger conditional probabilities.
The estimation of each conditional probability $\mathbb{P}(F_l|F_{l-1})$ requires sampling from the distribution of the random variables conditional on F_{l-1}, denoted as $Q(\cdot|F_{l-1})$, where $Q(\cdot|F_0)$ represents the initial input distribution and equals the generalized derivative of the input CDF $F_{\boldsymbol{X}}(\cdot)$. $Q(\cdot|F_0)$ can be sampled by standard Monte Carlo sampling, but the distributions $Q(\cdot|F_l), l > 0$, are only known point-wise up to a normalizing constant and, hence, cannot be sampled directly. Therefore MCMC sampling is employed. The sampling process in the l-th conditional sampling level is performed as follows: (1) Select the samples $\mathcal{P}^{(l-1)}$ from the $(l-1)$-th level that fall in F_l as the seeds $\mathcal{S}^{(l)}$ ($\mathcal{P}^{(0)}$ is generated through Monte Carlo sampling). (2) From each seed,

start a Markov chain that has the target distribution $Q(\cdot|F_l)$ as the stationary distribution, and record all the states as new samples $\mathcal{P}^{(l)}$. (3) Take the samples $\mathcal{P}^{(l)}$ located in F_{l+1} as new seeds $\mathcal{S}^{(l+1)}$ and estimate $\mathbb{P}(F_{l+1}|F_l)$ as $\frac{|\mathcal{S}^{(l+1)}|}{|\mathcal{P}^{(l)}|}$ where $|\mathcal{S}^{(l+1)}|$ and $|\mathcal{P}^{(l)}|$ denote the number of seeds and samples, respectively. The above three steps are repeated successively until F is approached. We note that the number of the samples per level $|\mathcal{P}^{(l)}|$ is usually fixed prior to the analysis.

Defining the intermediate events a priori is typically challenging. Hence, in standard SuS the intermediate failure events are chosen adaptively during the simulation such that each conditional probability equals a predefined constant p_0. This standard SuS approach is also termed (fixed effort) adaptive multilevel splitting [25]. In this variant, step (3) in the above sampling process is modified as follows: Order the samples $\mathcal{P}^{(l)}$ by their LSF values. The first p_0-percent of these sorted samples are then taken as seeds for the next sampling level and the LSF value of the p_0-percentile b_{l+1} is used to define the boundary of the next intermediate domain, such that $F_{l+1} = \{\boldsymbol{x} : g(\boldsymbol{x}) \leqslant b_{l+1}\}$. The resulting SuS estimator of the probability of failure is given as:

$$\widehat{p}_f = p_0^{m-1} \frac{N_f}{N} \tag{3}$$

where N and N_f represent the number of samples and failure samples at final level, respectively. The standard SuS algorithm is summarized in Algorithm 1.

As previously mentioned, MCMC sampling is applied to generate samples from each conditional distribution $Q(\cdot|F_l)$. In SuS, the seeds $\mathcal{S}^{(l)}$ already follow approximately the target distribution [23], hence, a burn-in period is not considered in practice. Since the samples generated from the same seed are states of the same Markov chain, they will be dependent; their correlation depends on the autocorrelation function of the underlying Markov chain. The stronger the correlation between samples, the larger the variance of the estimates of the conditional probabilities. Additionally, samples that share the same history, namely, their Markov chains are branches with root at the same Monte Carlo sample, will be correlated, even when they are in different levels. Such correlation introduces a dependency of the conditional probability estimators, which further increases the variance of the SuS estimator. Hence, the quality of the final probability estimate strongly depends on the particular choice and setting of the MCMC algorithm.

Algorithm 1: SuS algorithm

Input: $p_0 \in (0,1)$, an integer N multiple of $\frac{1}{p_0}$

$l \leftarrow 0$, $b_l \leftarrow \inf$

while $b_l > 0$ **do**

  if $l = 0$ **then**

    Generate N samples $\{\boldsymbol{x}_k\}_{k=1}^N$ from the initial distribution $Q(\cdot|F_0)$

  else

    Generate N samples $\{\boldsymbol{x}_k\}_{k=1}^N$ from the target distribution $Q(\cdot|F_l)$ with an MCMC algorithm with seeds $\mathcal{S}^{(l)}$

  Sort $\{\boldsymbol{x}_k\}_{k=1}^N$ by increasing order of their LSFs $g(\cdot)$, and denote the sorted samples as $\{\bar{\boldsymbol{x}}_k\}_{k=1}^N$

  $b_{l+1} \leftarrow g(\bar{\boldsymbol{x}}_{p_0 \cdot N})$

  if $b_{l+1} \leqslant 0$ **then**

    $b_{l+1} \leftarrow 0$

    $N_f = \sum_{k=1}^N \mathbb{I}\{g(\bar{\boldsymbol{x}}_k) \leqslant 0\}$

  Take the $\mathcal{S}^{(l+1)} \triangleq \{\bar{\boldsymbol{x}}_k\}_{k=1}^{p_0 \cdot N}$ as the seeds for the next level

  $l \leftarrow l + 1$

$\widehat{p}_f \leftarrow p_0^{l-1} \frac{N_f}{N}$

Output: $\widehat{p}_f$

2.2. Accuracy of the Subset Simulation estimator

The accuracy of the SuS estimator of the probability of failure $\widehat{p}_f$ can be assessed by the mean-square error, which is decomposed as:

$$\mathrm{MSE}(\widehat{p}_f) = (p_f - \mathbb{E}(\widehat{p}_f))^2 + \mathrm{Var}(\widehat{p}_f) \tag{4}$$

The first term on the right-hand side of Eq.(4) represents the bias contribution and the second term the variance of the SuS estimator.

Assume first that the intermediate events are defined before the simulation. In the Monte Carlo level ($l = 0$), samples $\mathcal{P}^{(0)}$ are generated from $Q(\cdot|F_0)$ independently, and therefore the seeds $\mathcal{S}^{(1)}$ follow the distribution $Q(\cdot|F_1)$. This will lead to so-called perfect sampling when simulating the Markov chains in the next level. Since the chains have already reached the stationary state at the beginning, no burn-in time is needed, and all samples $\mathcal{P}^{(1)}$ will follow $Q(\cdot|F_1)$. In this way, samples $\mathcal{P}^{(l)}$ generated in any l-th conditional level will follow the target distribution $Q(\cdot|F_l)$ and the corresponding estimator of the conditional probability $\widehat{p}(F_{l+1}|F_l)$ will be unbiased. Moreover, [25] proves that the resulting failure probability estimator $\widehat{p}_f$ is also unbiased if both intermediate events and length of the Markov chain are predefined, i.e., if they are independent of the simulation process.

Since the intermediate events are selected adaptively in SuS, samples $\mathcal{S}^{(l)}$ will not completely follow the target distribution. As a result, both conditional probability estimator and failure probability estimator will be slightly biased. Nevertheless, compared to the variance of the estimator, the squared bias is one order of magnitude smaller [23] and, hence, its contribution to the mean-square error (MSE) of the estimator is negligible. In other words, the error of the SuS is mainly due to the variance of the failure probability estimator rather than the bias. The most common and reliable way to calculate the variance $\mathrm{Var}(\widehat{p}_f)$ is to run SuS several times and to use the sample variance as the unbiased estimation of the $\mathrm{Var}(\widehat{p}_f)$. One can also evaluate the variance approximately through a single run of the SuS. More details can be found in [23] and [28]. However, this approximate estimator is shown to underrepresent the true variance of $\widehat{p}_f$, especially for small target p_f.

3. Markov Chain Monte Carlo algorithm for network reliability assessment

Most MCMC algorithms that are widely used in risk analysis can be regarded as variants of the Metropolis-Hastings (M-H) algorithm. These include, for example, Gibbs sampling [10][26], conditional sampling [28] and Hamiltonian Monte Carlo [36]. To sample from the intermediate target distribution $Q(\cdot|F_l)$ in SuS, M-H algorithm proceeds in the following two steps [28]:

1. Generate a candidate sample $\boldsymbol{v}$ from the distribution $Q_p^{(l)}(\cdot|\boldsymbol{x}_k)$ which is termed the proposal distribution.
2. Accept or reject $\boldsymbol{v}$.

$$\boldsymbol{x}_{k+1} = \begin{cases} \boldsymbol{v}, & \text{with prob.} \quad \alpha \\ \boldsymbol{x}_k, & \text{with prob.} \quad 1-\alpha \end{cases} \quad \text{where}$$

$$\alpha = \mathbb{I}\left\{\boldsymbol{v} \in F_l\right\} \min\left\{1, \frac{Q(\boldsymbol{v}|F_0)Q_p^{(l)}(\boldsymbol{x}_k|\boldsymbol{v})}{Q(\boldsymbol{x}_k|F_0)Q_p^{(l)}(\boldsymbol{v}|\boldsymbol{x}_k)}\right\}$$

It can be shown that the M-H algorithm satisfies the detailed balance condition independent of the choice of the proposal distribution. In this section, we first discuss the adaptive conditional sampling method of [28] in the context of network reliability assessment and then propose a more efficient yet less general independent M-H algorithm, which is applicable in problems with discrete input spaces.

3.1. Adaptive conditional sampling in standard normal space

3.1.1. Implementation in standard normal space

Let $\boldsymbol{U}$ denote an n-dimensional random vector that has the independent standard normal distribution. The original random vector $\boldsymbol{X}$ can be expressed in terms of the vector $\boldsymbol{U}$ through an isoprobabilistic mapping $\boldsymbol{T} : \mathbb{R}^n \to \mathbb{R}^n$. One can define the reliability problem in the $\boldsymbol{U}$-space as follows:

$$p_f = \mathbb{P}(g(\boldsymbol{X}) \leqslant 0) = \mathbb{P}(G(\boldsymbol{U}) \leqslant 0) = \int_{G(\boldsymbol{u}\leqslant 0)} \varphi_n(\boldsymbol{u}) \mathrm{d}\boldsymbol{u} \tag{5}$$

where $G(\boldsymbol{U}) = g(\boldsymbol{T}(\boldsymbol{U}))$ and $\varphi_n(\boldsymbol{u})$ is the n-dimensional independent standard normal joint probability density function (PDF). The mapping $\boldsymbol{T}(\cdot)$

can be obtained by the Rosenblatt transformation, which is implemented as follows:

$$\begin{aligned} x_1 &= F_{X_1}^{-1}(\Phi(u_1)) \\ x_2 &= F_{X_2}^{-1}(\Phi(u_2)|x_1) \\ &\vdots \\ x_n &= F_{X_n}^{-1}(\Phi(u_m)|x_1, \cdots, x_{n-1}) \end{aligned} \tag{6}$$

where Φ represents the CDF of standard normal distribution and $F_{X_d}(\cdot|x_1, \cdots, x_{d-1})$ denotes the conditional CDF of X_d given $X_1 = x_1, \cdots, X_{d-1} = x_{d-1}$. If any subset of $\boldsymbol{X}$ consists of discrete random variables, then it is possible that the functions $F_{X_d}(\cdot|x_1, \cdots, x_{d-1})$ are not strictly invertible. Therefore, we use the following extended definition of the inverse of a CDF

$$F^{-1}(a) = \inf(x : F(x) \geq a) \tag{7}$$

We note that in such cases the Rosenblatt transformation is not one-to-one and hence, the inverse mapping from $\boldsymbol{X}$ to $\boldsymbol{U}$ is not uniquely defined.

3.1.2. Adaptive conditional sampling algorithm

Having defined the transformation from the original $\boldsymbol{X}$-space to the $\boldsymbol{U}$-space, SuS can be used to solve the reliability problem in the transformed space. At the l-th level of SuS, MCMC sampling is applied to sample from the conditional standard normal density $p_{\boldsymbol{U}}(\cdot|F_l)$. Sampling according to this density can be performed by application of the aCS algorithm. Before describing the aCS algorithm, we first discus its non-adaptive variant, the standard conditional sampling (CS) algorithm. The transition from the current state $\boldsymbol{u}_k$ to a new state $\boldsymbol{u}_{k+1}$ using CS is as follows: First, a candidate $\boldsymbol{v}$ is generated. The CS sampler imposes that the candidate and the current state are jointly Gaussian with standard normal marginal distribution $\varphi_n(\cdot)$ and predefined symmetric cross-correlation matrix, $\boldsymbol{R}$. One then samples $\boldsymbol{v}$ from the joint Gaussian distribution conditional on the current state, i.e., from $\mathcal{N}(\boldsymbol{v}; \boldsymbol{R}\boldsymbol{u}_k, \boldsymbol{I} - \boldsymbol{R}\boldsymbol{R}^{\mathrm{T}})$, where $\boldsymbol{I}$ is the identity matrix. The candidate is accepted if it is located in the intermediate failure domain F_l, in which case it is set as the new state $\boldsymbol{u}_{k+1}$. Otherwise, $\boldsymbol{u}_k$ is taken as the new state. CS can be summarized as follows:

1. Generate candidate sample $\boldsymbol{v}$ from the normal distribution $\mathcal{N}(\boldsymbol{v}; \boldsymbol{R}\boldsymbol{u}_k, \boldsymbol{I} - \boldsymbol{R}\boldsymbol{R}^{\mathrm{T}})$.

2. Accept or reject $\boldsymbol{v}$.

$$\boldsymbol{u}_{k+1} = \begin{cases} \boldsymbol{v}, \boldsymbol{v} \in F_l \\ \boldsymbol{u}_k, \boldsymbol{v} \notin F_l \end{cases}$$

It can be proven that the above transition satisfies the detailed balance condition with respect to the target distribution $p_{\boldsymbol{U}}(\cdot|F_l)$, hence, $p_{\boldsymbol{U}}(\cdot|F_l)$ is the stationary distribution of the generated Markov chain [28, 37]. In fact, CS can be regarded as the M-H sampler with proposal distribution taken as $\mathcal{N}(\boldsymbol{v}; \boldsymbol{R}\boldsymbol{u}_k, \boldsymbol{I} - \boldsymbol{R}\boldsymbol{R}^{\mathrm{T}})$ [28]. We further note that the CS sampler will never generate repeated candidates, which results in an acceptance probability that is independent of the dimension of the vector $\boldsymbol{U}$. Hence, it is suitable for application to high-dimensional problems.

The performance of the CS sampler depends on the choice of the matrix $\boldsymbol{R}$ or, equivalently, the covariance matrix of the proposal distribution $\mathcal{N}$. Usually, $\boldsymbol{R}$ is chosen as a diagonal matrix with d-th diagonal term equal to ρ_d, which implies a component-wise sampling scheme, i.e., the d-th component of candidate, v_d, is sampled from $\mathcal{N}(v_d; \rho_d u_{k,d}, 1 - \rho_d^2)$. Large values of ρ_d lead to strong correlation between current and next state, but small values also lead to increased correlation due to the high rejection rate in the second step of CS. The correlation of the generated Markov chains can be controlled by choosing ρ_d or, equivalently, the standard deviation $\sigma_d = \sqrt{1 - \rho_d^2}$ adaptively, employing intermediate results from the simulation. In adaptive MCMC algorithms, the chain correlation is usually controlled by matching a near-optimal acceptance probability of the chain [38]. The aCS algorithm [28] adapts the sampling parameters by running batches of N_a chains starting from randomly selected seeds. After running each batch, the acceptance probability is estimated and the sampling parameters are adapted to match a pre-defined acceptance probability α^*. The aCS algorithm for generating N samples according to $p_{\boldsymbol{U}}(\cdot|F_l)$ starting from seeds $\mathcal{S}^{(l)}$ is given in Appendix A.

3.2. Independent Metropolis-Hastings algorithm

In network reliability assessment, the probability content at the intermediate domains in SuS typically centers at multiple discrete system states (modes) and, hence, the intermediate target distribution is multimodal. To efficiently sample from such distribution, we propose a novel independent M-H algorithm. The algorithm is applicable when all input random variables are discrete and exploits the information of the discarded samples in previous

sampling levels to form a proper proposal distribution in the M-H algorithm that is independent of the current state of the chain. Specifically, let $\mathcal{X}_l$ represent the set of samples discarded at level l, in other words, the generated samples at level l that are not in the $(l+1)$-th intermediate domain. The proposal distribution $Q_p^{(l)}(\boldsymbol{x})$ of the M-H algorithm at level l is then defined as the original input distribution $Q(\boldsymbol{x}|F_0)$ excluding the states visited by the previously discarded samples. That is,

$$Q_p^{(l)}(\boldsymbol{x}) \propto Q(\boldsymbol{x}|F_0)\mathbb{I}\left\{\boldsymbol{x} \notin \cup_{i=0,\ldots,l-1}\mathcal{X}_i\right\} \tag{8}$$

Since all samples in $\cup_{i=0,\ldots,l-1}\mathcal{X}_i$ are located outside the intermediate domain F_l, F_l is included in the support of the proposal distribution, $\Omega_p^{(l)} = \boldsymbol{x} \notin \cup_{i=0,\ldots,l-1}\mathcal{X}_i$. This is illustrated in Fig. 2. Moreover, the proposal distribution

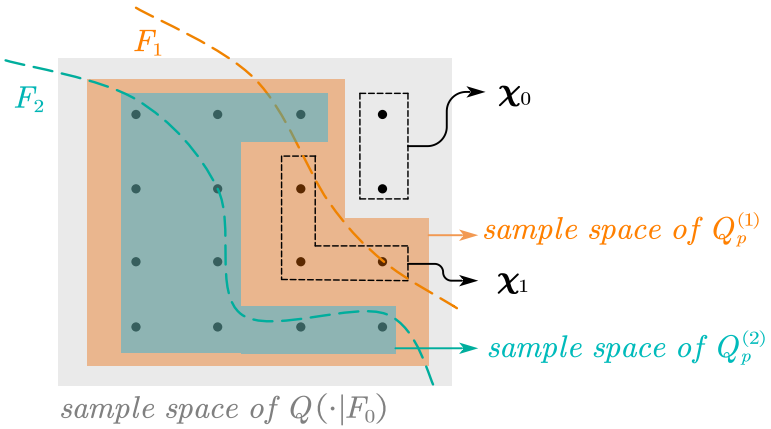


Figure 2: Schematic diagram of proposal distributions. (The black dots represent the basic random events, and the dotted curve indicates the intermediate failure domain)

has exactly the same shape as the intermediate target distribution in F_l as they are both proportional to the input distribution $Q(\boldsymbol{x}|F_0)$. The acceptance rate α of the candidate generated by this proposal is given as

$$\alpha = \mathbb{I}\left\{\boldsymbol{x} \in F_l\right\} \min\left\{1, \frac{Q(\boldsymbol{x}|F_0)Q(\boldsymbol{x}_k|F_0)}{Q(\boldsymbol{x}_k|F_0)Q(\boldsymbol{x}|F_0)}\right\} = \mathbb{I}\left\{\boldsymbol{x} \in F_l\right\} \tag{9}$$

Note that both the generation and the acceptance of the candidate are independent of the current state. In the literature, M-H samplers whose proposal distribution is independent of the current state are termed independent M-H samplers. We note that the samples generated by this algorithm are not

independent, since we get a repeated sample when rejecting the candidate. This is the main difference between the independent M-H algorithm and rejection sampling with envelope $\frac{1}{\mathbb{E}[\alpha]}Q_p^{(l)}(\boldsymbol{x})$, which generates independent samples [39]. In the latter, one gets a new sample only when the candidate is accepted and, hence, the computational cost of rejection sampling is much higher than the independent M-H algorithm especially as the mean acceptance rate $\mathbb{E}[\alpha]$ is small. The average acceptance probability, $\mathbb{E}[\alpha]$, can be calculated through dividing $\mathbb{P}(\boldsymbol{X} \in F_l)$ by $\mathbb{P}(\boldsymbol{X} \notin \cup_{i=0,\dots,l-1}\mathcal{X}_i)$, i.e.,

$$\mathbb{E}_{Q_p^{(l)}}[\alpha] = \frac{\mathbb{P}(\boldsymbol{X} \in F_l)}{\mathbb{P}(\boldsymbol{X} \notin \cup_{i=0,\dots,l-1}\mathcal{X}_i)} \tag{10}$$

The magnitude of $\mathbb{E}[\alpha]$ tends to decrease as the intermediate level goes higher. Additionally, the sample size at each level, the dimension of the problem and the input distribution will also influence the mean acceptance rate $\mathbb{E}[\alpha]$. The influence is investigated in detail through a binomial experiment in Section 5.

The implementation of the above independent M-H algorithm is relatively simple and can be performed in the following two steps:

1. Generate candidate sample $\boldsymbol{v}$ from the proposal distribution $Q_p^{(l)}(\boldsymbol{x})$.
2. Accept or reject $\boldsymbol{v}$.

$$\boldsymbol{x}_{k+1} = \begin{cases} \boldsymbol{v}, \boldsymbol{v} \in F_l \\ \boldsymbol{x}_k, \boldsymbol{v} \notin F_l \end{cases}$$

To sample from the proposal distribution, $Q_p^{(l)}(\boldsymbol{x})$, one can sample directly from the input distribution and keep those samples that differ from the previous discarded samples. However such process can be quite inefficient when the proposal distribution is far from the input distribution, for instance, at deep intermediate levels. Such issue can be circumvented by applying the bound-based sampling algorithm [12], given that the input random variables are multivariate categorical distributed. Details on this algorithm can be found in Appendix B.

4. Adaptive effort subset simulation method

In each conditional level l of the SuS method with fixed number of samples per level and adaptive estimation of the intermediate events, the p_0-percentile of the LSF values of the samples $\mathcal{P}^{(l)}$, b_{l+1}, is used to define the boundary

of the intermediate domain. This adaptive approach works well when only a few samples are located on the boundary $g(\boldsymbol{x}) = b_{l+1}$, i.e., a few samples have the same LSF value as the p_0-percentile. However, it can happen that many samples fall on this boundary, particularly in the following cases:

(1) $\boldsymbol{X}$ includes discrete random variables.
(2) The LSF is defined such that the probability measure of the set $\{\boldsymbol{x} : g(\boldsymbol{x}) = b_{l+1}\}$ is strictly greater than zero.
(3) The parameters of the MCMC algorithm are inappropriately set, resulting in the candidates being rejected successively many times.

While case (3) can be avoided by an appropriate implementation of the MCMC algorithm, cases (1) and (2) are common in the context of network reliability assessment. This will result in an ambiguous definition of the intermediate domain F_{l+1} and can lead to an inaccurate estimate of the failure probability. In extreme situations, all samples generated in a certain level will have the same LSF value and the adaptive sampling process can get stuck and never reach the failure domain.
In this section, we modify the standard SuS algorithm to circumvent this problem. The resulting algorithm modifies the adaptive selection of the intermediate domains and adapts the number of samples per level (sampling effort) throughout the simulation. We term the proposed approach aE-SuS. As will be made clear, these modifications enable application of the method to general network reliability problems. The proposed algorithm is introduced in the following and summarized in Algorithm 2.

4.1. Intermediate domains

In order to provide a clear (unambiguous) definition of the intermediate domains, one can apply the following adaptive approach. At each conditional level, generate a set of samples $\mathcal{P}^{(l)}$ and define a temporary event F_{temp} as $\{\boldsymbol{x} : g(\boldsymbol{x}) \leqslant b_{l+1}\}$ where b_{l+1} is the p_0-percentile of the LSF values of $\mathcal{P}^{(l)}$. If $F_{temp} = F_l$, define the next intermediate event F_{l+1} as $\{\boldsymbol{x} : g(\boldsymbol{x}) < b_{l+1}\}$, otherwise set $F_{l+1} = F_{temp}$. This approach guarantees that $F_{l+1} \subsetneq F_l$, which avoids a degeneracy of the sampling process.
Because of the discrete nature of $g(\boldsymbol{x})$, it might be difficult to check whether $F_{temp} = F_l$ or not when $F_l = \{\boldsymbol{x} : g(\boldsymbol{x}) < b_l\}$. Therefore, we check if $b_{l+1} = \max\{g(\boldsymbol{x}) : \boldsymbol{x} \in \mathcal{P}^{(l)}\}$ instead. The latter condition checks whether the p_0 percentile of the LSF values equals the maximum LSF value of the samples $\mathcal{P}^{(l)}$, and is a necessary (but not sufficient) condition of $F_{temp} = F_l$.

Note that $F_{l+1} \subsetneq F_l$ still remains true after this modification, since $F_l / F_{l+1} \neq \emptyset$ and contains at least the samples taking the maximum LSF value in $\mathcal{P}^{(l)}$. The above adaptive approach for choosing the intermediate domains is described in lines 11-20 of Algorithm 2.

4.2. Sampling at the intermediate levels

The approach for selecting the intermediate domains, introduced in Section 3.1, could potentially lead to a very small number of failure samples per level, which reduces the accuracy of the estimates of the intermediate conditional probabilities. We hence need to adapt the number of samples per level to ensure that these estimates remain accurate. We first calculate the number of samples that fall into the domain F_{l+1} (the number of seeds N_s). If this number is smaller than a predefined constant C, we increase the current sampling effort N_c and append $\mathcal{P}^{(l)}$ with N new samples. Denote the extended sample set as $\mathcal{P}^{(l)}_{\text{ext}}$. The new samples should also follow the target distribution $Q(\cdot|F_l)$, and hence approximately $\mathbb{P}(F_{l+1}|F_l)$ of these samples should be located in F_{l+1}. Therefore, one needs to further generate $N = N_c \cdot \frac{C-N_s}{N_s}$ samples to get approximately C seeds, which is shown in line 22 of Algorithm 2. Note that $N > 0$ always holds. In the algorithm, the constant C is taken as a predefined proportion $tol \in (0, 1)$ of $p_0 \cdot N_0$, the product of initial conditional probability and the initial sample size. Larger tol will lead to more accurate but less efficient result. We have found $tol \in (0.5, 0.8)$ to be a good choice for the investigated cases.

In practice, the above appending process may need to be iterated several times to achieve at least C seeds. For a fixed F_{l+1}, with every iteration and increasing number of samples, the number of the seeds will keep increasing until the desired threshold C is achieved. By doing this, even in the extreme case where all the samples in $\mathcal{P}^{(l)}$ have the same LSF value, the sampling process will keep moving forward towards the failure domain and will no longer get stuck in this level as in the standard SuS algorithm.

To append new samples which follow the target distribution $Q(\cdot|F_l)$, we propose to extend the Markov chains generated in the initial intermediate sampling step ($iter = 0$, in Algorithm 2). This is illustrated in Fig. 3. As shown in the figure, for each seed in $\mathcal{S}^{(l)}$, a Markov chain is constructed in the 0th iteration. The last sample (tail) of this chain is then taken as the seed for the chain in the next iteration. The transition distribution of the chain remains unchanged. The above process may be iterated several times and is described as Algorithm 3. For iteration $it = 0, \cdots$, the input of Algorithm 3 consists

of 4 values: the number of samples to append N, the number of Markov chains $N_{ch} = |\mathcal{S}^{(l)}|$, the transition distribution of each chain $\{\Gamma_i\}_{i=1}^{N_{ch}}$, which is determined by the target distribution $Q(\cdot|F_l)$ and the employed MCMC algorithm, and the seed for each chain $\{\boldsymbol{e}_i^{(it)}\}_{i=1}^{N_{ch}}$, which is taken as $\mathcal{S}^{(l)}$ when $it = 0$ and otherwise as the tail of the Markov chains from the previous iteration $\{\boldsymbol{t}_i^{(it-1)}\}_{i=1}^{N_{ch}}$. The output of the algorithm is N new generated samples $\mathcal{P}_{\text{new}}^{(l)}$ and the tail (last sample) of each chain $\{\boldsymbol{t}_i^{(it)}\}_{i=1}^{N_{ch}}$.

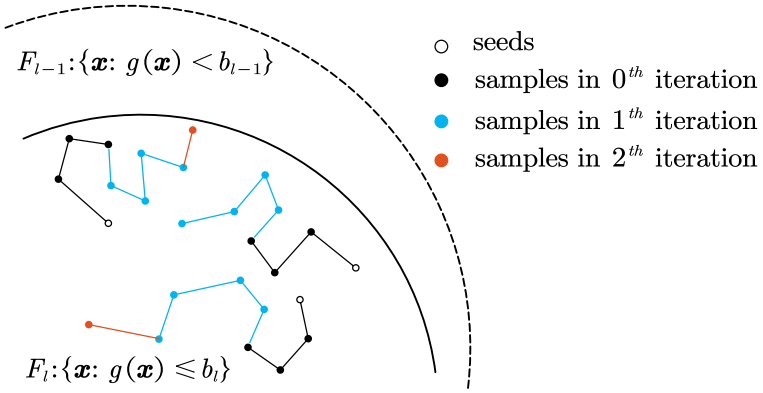


Figure 3: Schematic diagram of the appending method.

5. Examples

5.1. Multistate random variable

Consider a discrete random variable X with 7 states $\{x_1, \cdots, x_7\}$. We consider two cases. In case 1, the CDF of X , $F_X(\cdot)$, is set such that $\frac{F_X(x_{i+1})}{F_X(x_i)} \leq 10$, while in case 2, there is a big 'jump' between the third and the fourth state, i.e., $\frac{F_X(x_4)}{F_X(x_3)} \approx 599$. The CDF of X for the two considered cases is given in Table 1 and is illustrated in Fig 4. The LSF is defined as $g(X) = X + 5$ such that the failure probability $\mathbb{P}(X \leqslant -5)$ equals 10^{-5} for both cases. We implement SuS and the proposed aE-SuS respectively in standard normal space to evaluate the failure probability and compare them with crude MCS. The MCMC algorithm is aCS. For SuS, the sampling effort is fixed to 1,000, and the conditional probability is 0.1. For aE-SuS, the parameters are set to be $tol = 0.5, N_0 = 1{,}000, p_0 = 0.1$. Each method is run 1,000 times to get the relative bias, coefficient of variation and average computational cost of the

Algorithm 2: Adaptive effort subset simulation algorithm

Input: $tol \in (0,1)$, $p_0 \in (0,1)$, an integer N_0 multiple of $1/p_0$

$l \leftarrow 0, b_l \leftarrow \inf, N \leftarrow N_0, \mathcal{P}^{(l)} \leftarrow \emptyset$

while $b_l > 0$ **do**

  $iter \leftarrow 0$, $N_s \leftarrow 0$

  while $N_s < tol \cdot N_0 \cdot p_0$ **do**

    if $l = 0$ **then**

      Generate N samples $\{\boldsymbol{x}_k\}_{k=1}^{N}$ from the initial distribution $Q(\cdot|F_0)$ and add them to $\mathcal{P}^{(l)}$

    else

      Generate N samples $\{\boldsymbol{x}_k\}_{k=1}^{N}$ from the target distribution $Q(\cdot|F_l)$ with appending algorithm and add them to $\mathcal{P}^{(l)}$

    $N_c \leftarrow |\mathcal{P}^{(l)}|$ // total sample size

    Sort the elements of $\mathcal{P}^{(l)}$ by increasing order of their LSF values $g(\boldsymbol{x})$, and denote the sorted samples as $\{\bar{\boldsymbol{x}}_k\}_{k=1}^{N_c}$

    if $iter = 0$ **then**

      $b_{l+1} \leftarrow g(\bar{\boldsymbol{x}}_{p_0 \cdot N_0})$

      if $b_{l+1} \leqslant 0$ **then**

        $b_{l+1} \leftarrow 0$

        $N_s \leftarrow \sum_{k=1}^{N_c} \mathbb{I}\{g(\bar{\boldsymbol{x}}_k) \leqslant b_{l+1}\}$, $F_{l+1} \triangleq \{\boldsymbol{x} : g(\boldsymbol{x}) \leqslant b_{l+1}\}$

        Break

      else if $b_{l+1} < g(\bar{\boldsymbol{x}}_{N_c})$ **then**

        $N_s \leftarrow \sum_{k=1}^{N_c} \mathbb{I}\{g(\bar{\boldsymbol{x}}_k) \leqslant b_{l+1}\}$, $F_{l+1} \triangleq \{\boldsymbol{x} : g(\boldsymbol{x}) \leqslant b_{l+1}\}$

        Break

    $N_s \leftarrow \sum_{k=1}^{N_c} \mathbb{I}\{g(\bar{\boldsymbol{x}}_k) < b_{l+1}\}$, $F_{l+1} \triangleq \{\boldsymbol{x} : g(\boldsymbol{x}) < b_{l+1}\}$

    if $N_s < tol \cdot N_0 \cdot p_0$ **then**

      $N \leftarrow \lceil N_c \cdot \frac{tol \cdot N_0 \cdot p_0}{max(1, N_s)} \rceil - N_c \geqslant 1$

      $iter \leftarrow iter + 1$

  Take the $\mathcal{S}^{(l+1)} \triangleq \{\bar{\boldsymbol{x}}_k\}_{k=1}^{N_s}$ as the seeds for the next level

  $N \leftarrow N_0 - N_s$, $\mathcal{P}^{(l+1)} \leftarrow \mathcal{S}^{(l+1)}$, $l \leftarrow l + 1$

  $\widehat{p}(F_l|F_{l-1}) \leftarrow \frac{N_s}{N_c}$

$\widehat{p}(F) \leftarrow \prod_{j=1}^{l} \widehat{p}(F_j|F_{j-1})$

Output: $\widehat{p}(F)$

Algorithm 3: Appending algorithm

Input: N, N_{ch}, $\{\Gamma_i\}_{i=1}^{N_{ch}}$, $\{\boldsymbol{e}_i^{(it)}\}_{i=1}^{N_{ch}}$

Randomly choose $\mathrm{mod}(N, N_{ch})$ elements from the set $\{1, 2, \cdots, N_{ch}\}$, say $\boldsymbol{\chi}$

$\mathcal{P}_{\text{new}}^{(l)} \leftarrow \emptyset$

for $i = 1, \cdots, N_{ch}$ **do**

  if $i \in \boldsymbol{\chi}$ **then**

    $j \leftarrow \lfloor \frac{N}{N_{ch}} \rfloor + 1$

  else

    $j \leftarrow \lfloor \frac{N}{N_{ch}} \rfloor$

  $\boldsymbol{x}_0 \leftarrow \boldsymbol{e}_i^{(it)}$

  for $k = 1, \cdots, j$ **do**

    Sample $\boldsymbol{x}_k$ from transition density $\Gamma_i(\cdot|\boldsymbol{x}_{k-1})$

    Add $\boldsymbol{x}_k$ to $\mathcal{P}_{\text{new}}^{(l)}$

  $\boldsymbol{t}_i^{(it)} \leftarrow \boldsymbol{x}_j$

Output: $\mathcal{P}_{\text{new}}^{(l)}$, $\{\boldsymbol{t}_i^{(it)}\}_{i=1}^{N_{ch}}$

failure probability estimator. The results for case 1 and case 2 are shown in Tables 2 and 3 respectively. In both cases, aE-SuS shows good accuracy, a negligible bias and a much smaller coefficient of variation than crude MCS. We note that the coefficient of variation of crude MCS is given for the same computational effort as the proposed aE-SuS method. In contrast, SuS gives a strongly biased estimate of the failure probability with high coefficient of variation in the first case and falls into a dead loop in the second case.

Table 1: CDF of X for Example 5.1.

State	-6	-4	-3	-2	-1	0	1
CDF(case1)	1e-5	1e-4	1e-3	1e-2	1e-1	5e-1	1
CDF(case2)	1e-5	3e-5	5e-5	3e-2	1e-1	5e-1	1

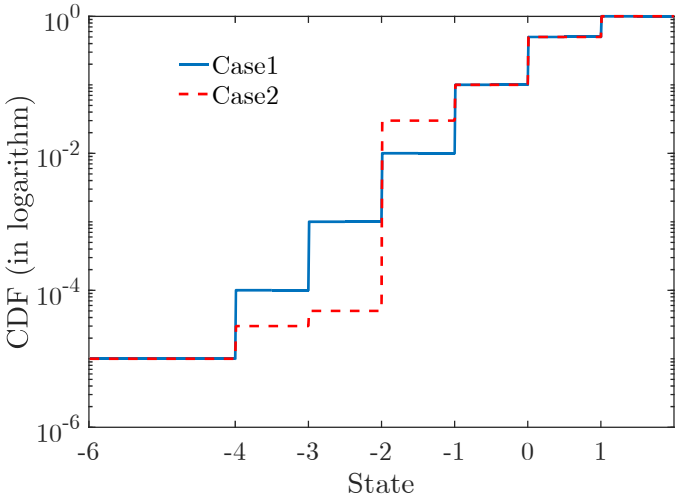


Figure 4: CDF of X for Example 5.1.

Table 2: Statistical characteristics of the estimator of the probability of Example 5.1 (Case 1).

	relative bias(%)	coefficient of variation	average computational effort
SuS	-97.8	3.747	7,222
aE-SuS	3.5	0.376	5,970
MCS	0	4.093	5,970

Table 3: Statistical characteristics of the estimator of the probability of Example 5.1 (Case 2).

	relative bias(%)	coefficient of variation	average computational effort
SuS	/	/	/
aE-SuS	2.4	0.242	44,737
MCS	0	1.495	44,737

5.2. Multidimensional flow-based problem

In this example, the failure event is defined as

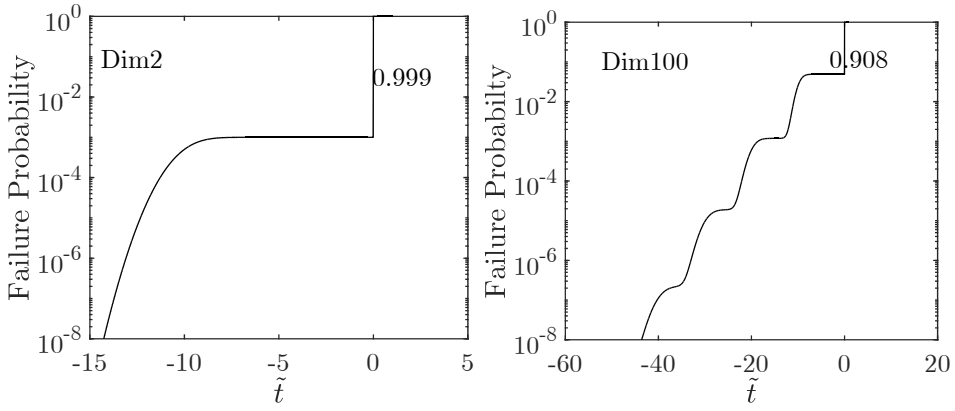
$$\sum_{i=1}^{n} X_i(1-Y_i) \geqslant t \tag{11}$$

where X_i are independent and identically distributed (iid) according to the normal distribution $\mathcal{N}(\cdot;\mu,\sigma^2)$ and Y_i are also iid and follow the Bernoulli distribution $\mathrm{Ber}(1-p_{fc})$ with outcomes $\{0,1\}$. Each X_i can be regarded as a loss variable associated to a failure event with probability p_{fc}. The failure event is further defined as the total loss exceeding the predefined threshold t. Let $\widetilde{X}_i = -X_i \sim \mathcal{N}(\cdot;-\mu,\sigma^2)$, $\widetilde{t} = -t$, and $\widetilde{Y}_i = 1-Y_i \sim \mathrm{Ber}(p_{fc})$. The failure probability then becomes

$$\begin{aligned} p_f &= \mathbb{P}\left(\sum_{i=1}^{n} -X_i(1-Y_i) \leqslant -t\right) = \mathbb{P}\left(\sum_{i=1}^{n} \widetilde{X}_i\widetilde{Y}_i \leqslant \widetilde{t}\right) \\ &= \sum_{i=0}^{n} \mathbb{P}\left(\sum_{j=1}^{n} \widetilde{Y}_j = i\right) \mathbb{P}\left(\sum_{k=1}^{n} \widetilde{X}_k\widetilde{Y}_k \leqslant \widetilde{t} \,\middle|\, \sum_{j=1}^{n} \widetilde{Y}_j = i\right) \\ &= \sum_{i=1}^{n} \tbinom{n}{i} p_{fc}^{i}(1-p_{fc})^{n-i}\Phi\left(\frac{\widetilde{t}-(-\mu i)}{\sqrt{i\cdot\sigma^2}}\right) + (1-p_{fc})^n \mathbb{I}(\widetilde{t} \geqslant 0) \end{aligned} \tag{12}$$

Eq.(12) shows that the failure probability p_f is a function of n and $\widetilde{t}$ when fixing μ, σ and p_{fc}. For different n, we choose $\widetilde{t}$ such that the failure probability equals to p_f^*. In this way, we define a series of failure events of different dimension but with the same failure probability, p_f^*. Note that there is a 'jump' of value $(1-p_{fc})^n$ at the origin coordinate; this value decreases as the dimension increases. Fig. 5 shows the failure probability p_f as a function of $\widetilde{t}$ for the case $n = 1$ (the dimension is 2) and $n = 50$ (the dimension is 100). The failure probability can also be regarded as the CDF of $\sum_{i=1}^{n} \widetilde{X}_i\widetilde{Y}_i$.

Next, MCS, standard SuS and aE-SuS are carried out to obtain the failure probabilities. SuS and aE-SuS are performed in standard normal space with aCS as the MCMC algorithm. Here, we set $p_{fc} = p_f^* = 10^{-3}, \mu = -10, \sigma = 1$ and vary n from 1 to 50. tol, N_0 and p_0 for aE-SuS are set to be 0.5, 1,000, 0.1, respectively. As shown in Fig. 6, the computational cost of aE-SuS, which is measured by the total number of LSF evaluations, decreases rapidly with increasing dimension and reaches around 4,000 calls of the LSF for higher



(a) $n = 1$ (b) $n = 50$

Figure 5: Failure probability p_f vs. $\widetilde{t}$ for the flow problem of Example 5.2.

dimensions. In order to obtain the statistical characteristics of the aE-SuS estimator and to compare them with MCS and standard SuS, 500 independent trials of aE-SuS and SuS are carried out. The results of MCS are calculated theoretically with the same computational cost (total number of samples N_{tot}) as aE-SuS. The MCS estimator is unbiased and the coefficient of variation is $\frac{1-p_f}{\sqrt{N_{tot} \cdot p_f}}$. Fig. 7 and Fig. 8 illustrate the relative bias and the coefficient of variation of both aE-SuS and MCS. We see that the behavior of aE-SuS is similar to that of MCS in low dimensions where the jump at the origin is large, while in high dimensions where the jump of the CDF becomes smaller, aE-SuS is more efficient. We note that the influence of the number of random variables on the performance of the MCMC algorithm used in aE-SuS is insignificant. This is due to the fact that the aCS is especially designed for high dimensional problems.

Fig. 9 compares the square root of MSE (RMSE, calculated through estimates of the two terms of Eq.(4)) of aE-SuS with different settings of standard SuS. It can be seen that even with well-tuned parameters ($p_0 = 1/40$), standard SuS can lead to significant errors in low to moderate dimensions where the jump in the CDF of the LSF is large.

As the 'jump' vanishes for large n, the results of aE-SuS become similar to that of standard SuS. In low dimensions, the aE-SuS algorithm behaves similar to crude MCS.

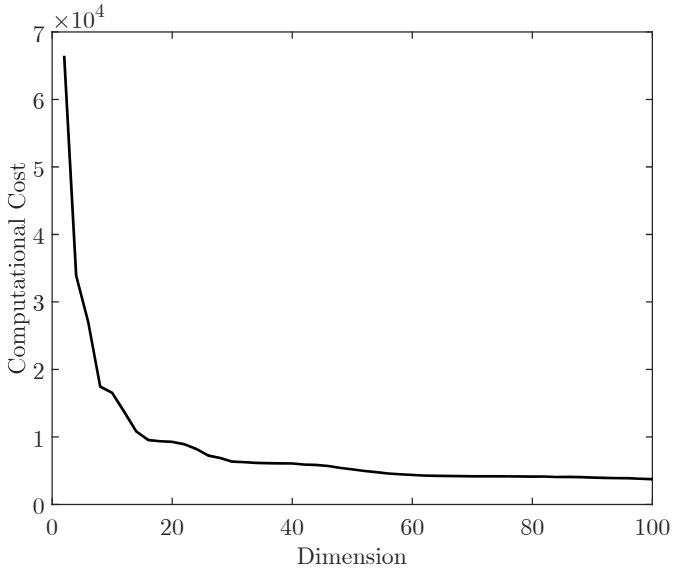


Figure 6: Computational cost of the aE-SuS for Example 5.2.

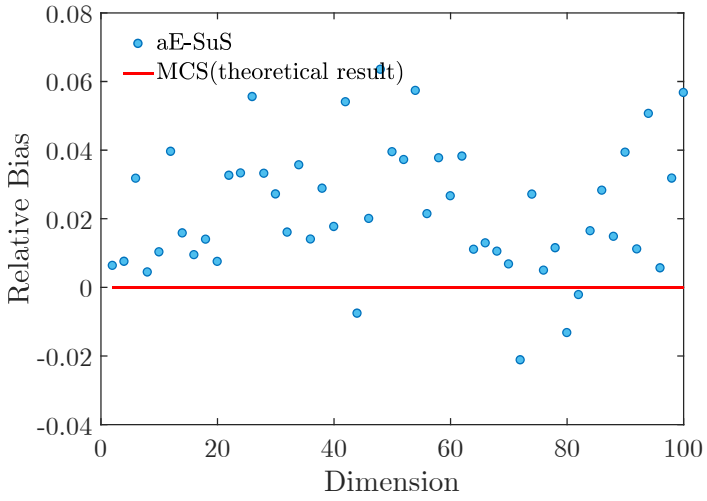


Figure 7: Relative bias (aE-SuS vs. MCS) for Example 5.2.

5.3. Binomial experiment

This example studies the behavior of the independent M-H algorithm proposed in Section 3.2. Consider a binomial experiment with n trials. Each trial is an independent event that has two outcomes: 0 and 1. The probability that a trial is successful (takes outcome 1) is equal to p and we evaluate the

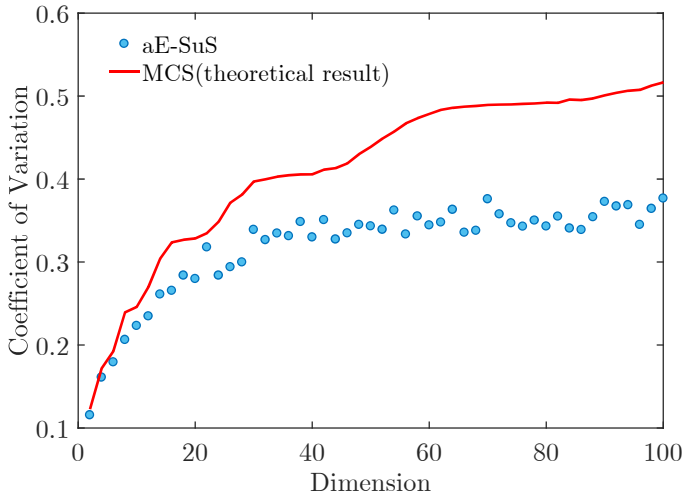


Figure 8: Coefficient of variation (aE-SuS vs. MCS) for Example 5.2.

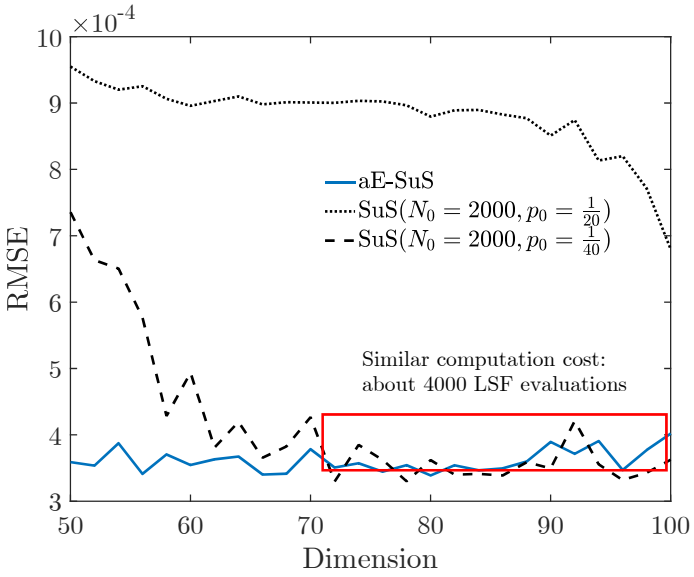


Figure 9: RMSE (aE-SuS vs. SuS) for Example 5.2.

probability that at least t trials are successful. The LSF is then defined as:

$$g(\boldsymbol{X}) = t - \sum_{i=1}^{n} X_i$$

where X_i represents the outcome of the i-th trial. The exact failure probability can be expressed as $1 - F_B(t-1; n, p_{f_c})$ where $F_B(\cdot; n, p)$ is the CDF of the binomial distribution with parameters n and p.

In order to study the performance of the independent M-H algorithm in different dimensions, we fix p at 10^{-3} and vary n. For each n, aE-SuS with the independent M-H algorithm is run 200 times with parameters $tol = 0.8, p_0 = 0.1, N_0 = 2{,}000$.

For different dimensions and conditional levels of aE-SuS, the mean acceptance rate of the independent M-H algorithm, $\mathbb{E}(\alpha)$, is calculated through Eq.(10) and is summarized in Table 4. Note that $\mathbb{P}(\boldsymbol{X} \in F_l)$ and $\mathbb{P}(\boldsymbol{X} \notin \cup_{i=0,\dots,l-1}\mathcal{X}_i)$ in Eq.(10) are abbreviated as $\mathbb{P}(F_l)$ and $\mathbb{P}(\overline{\cup\mathcal{X}_i})$, respectively. One can see that both $\mathbb{P}(F_l)$ and $\mathbb{P}(\overline{\cup\mathcal{X}_i})$ decrease as conditional level l increases. However, $\mathbb{P}(\overline{\cup\mathcal{X}_i})$ drops much slower than $\mathbb{P}(F_l)$ at high levels, which results in small $\mathbb{E}(\alpha)$. This is because, to form a good proposal in the independent M-H algorithm, the states that need to be excluded from the input distribution grow exponentially with l. This effect is more pronounced in high dimensions, leading to a faster decrease of $\mathbb{E}(\alpha)$. At the fourth level, the mean acceptance rate for $n = 100$ is only 0.2% of the rate for $n = 25$ in this example.

Nevertheless, if the mean acceptance rate is not too small, the independent M-H algorithm performs well. For instance, if we fix the threshold t at 4, the results of 200 independent runs of aE-SuS are summarized in Table 5. For all 4 cases, aE-SuS is slightly biased with less computational cost than crude MCS for achieving the same coefficient of variation. Note that the average computational cost for $n = 25$ is much higher than the other three cases, which is due to the larger 'jumps' in the CDF of the LSF.

5.4. Power network system

In this example, we consider the IEEE39 bus benchmark system, which consists of 39 nodes and 46 weighted edges. The topology of the network is illustrated in Fig. 10 where orange circles represent the source nodes and grey circles represent the terminal nodes. Edges are weighted by their reactance values shown on the right-hand side of Fig. 10 and by their capacities shown on the left-hand side. The line capacity is modeled here as being proportional to the number of most efficient paths between any source and terminal node pair passing through that line. This example was previously investigated by Scherb et al. [40] to quantify the network reliability considering cascading

Table 4: Mean acceptance rate of the independent M-H algorithm for Example 5.3.

		$n = 25$	$n = 50$	$n = 75$	$n = 100$
$l = 1$ $(g(\boldsymbol{X}) < t)$	$\mathbb{P}(F_l)$	0.025	0.049	0.072	0.095
	$\mathbb{P}(\overline{\cup \mathcal{X}_i})$	0.025	0.049	0.072	0.095
	$\mathbb{E}(\alpha)$	1	1	1	1
$l = 2$ $(g(\boldsymbol{X}) < t - 1)$	$\mathbb{P}(F_l)$	$2.95 \cdot 10^{-4}$	0.0012	0.0026	0.0046
	$\mathbb{P}(\overline{\cup \mathcal{X}_i})$	$2.95 \cdot 10^{-4}$	0.0012	0.0026	0.0046
	$\mathbb{E}(\alpha)$	1	1	1	1
$l = 3$ $(g(\boldsymbol{X}) < t - 2)$	$\mathbb{P}(F_l)$	$2.26 \cdot 10^{-6}$	$1.89 \cdot 10^{-5}$	$6.40 \cdot 10^{-5}$	$1.50 \cdot 10^{-4}$
	$\mathbb{P}(\overline{\cup \mathcal{X}_i})$	$2.26 \cdot 10^{-6}$	$1.89 \cdot 10^{-5}$	$1.87 \cdot 10^{-4}$	0.0017
	$\mathbb{E}(\alpha)$	1	1	0.34	0.087
$l = 4$ $(g(\boldsymbol{X}) < t - 3)$	$\mathbb{P}(F_l)$	$1.24 \cdot 10^{-8}$	$2.22 \cdot 10^{-7}$	$1.15 \cdot 10^{-6}$	$3.63 \cdot 10^{-6}$
	$\mathbb{P}(\overline{\cup \mathcal{X}_i})$	$1.24 \cdot 10^{-8}$	$9.70 \cdot 10^{-6}$	$1.84 \cdot 10^{-4}$	0.0017
	$\mathbb{E}(\alpha)$	1	0.023	0.0063	0.0021

Table 5: Statistics of the aE-SuS estimator for Example 5.3.

	$n = 25$	$n = 50$	$n = 75$	$n = 100$
p_f	$1.24 \cdot 10^{-8}$	$2.22 \cdot 10^{-7}$	$1.15 \cdot 10^{-6}$	$3.63 \cdot 10^{-6}$
relative bias(%)	2	3	5	2
coefficient of variation	0.15	0.14	0.20	0.34
average cost	$8.23 \cdot 10^4$	$3.77 \cdot 10^4$	$2.68 \cdot 10^4$	$2.09 \cdot 10^4$
MCS cost	$3.58 \cdot 10^9$	$2.30 \cdot 10^8$	$2.17 \cdot 10^7$	$2.38 \cdot 10^6$

effects and spatially distributed hazards, and by ro-Velasquez and Straub [41] to select representative failure scenarios.

The state of each node is considered as an independent Bernoulli random

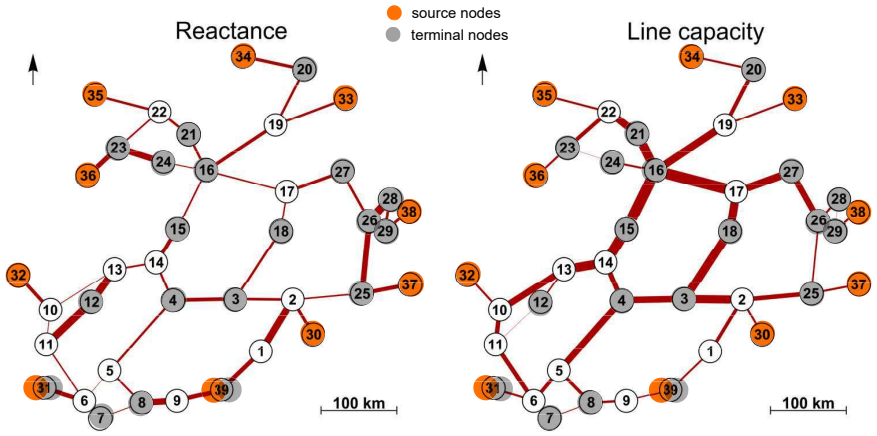


Figure 10: IEEE39 bus system, with edge thicknesses proportional to their estimated capacities (left) and reactances (right)[40].

variable, with component failure probability randomly chosen from the uniform distribution $U[0, 10^{-2}]$. The LSF is then defined as a function of the system state $\boldsymbol{x}$, which is a binary vector, as follows:

$$g(\boldsymbol{x}) = \frac{E(\boldsymbol{x})}{E(\mathbf{1})} - \text{threshold} \tag{13}$$

$$E(\boldsymbol{x}) = \frac{1}{|SN||TN|} \sum_{s \in SN, t \in TN, t \neq s} eff_{st}(\boldsymbol{x}) \tag{14}$$

eff_{st} is the efficiency of the most efficient path from source node s to terminal node t, which is evaluated as the inverse of the sum of the reactance values along that path. $E(\boldsymbol{x})$ is the efficiency of the whole system associated to the system state $\boldsymbol{x}$ (The vector $\mathbf{1}$ is the intact system state). It is equal to the mean value of all the eff_{st} from each source node in set SN to each terminal nodes in set TN.

In order to model cascading effects, Eq.(13) is modified to

$$g(\boldsymbol{x}) = \frac{E(\mathcal{C}(\boldsymbol{x}))}{E(\mathbf{1})} - \text{threshold} \tag{15}$$

where $\mathcal{C}(\boldsymbol{x})$ is the final system state after cascading effects due to overloading of system components. These are triggered by overloading in individual lines following initial failures, and are modeled following [40] and [42].
The threshold is fixed to 0.3, which means the system is considered as failed when its efficiency is less than 30% of that of the intact system. We apply the aE-SuS algorithm in original Bernoulli space and set the parameters $N = 2{,}000, p_0 = 0.1, tol = 0.8$. The MCMC algorithm is the independent M-H algorithm. Fig. 11 shows the empirical CDF of $g(\boldsymbol{X})$ obtained by MCS and the aE-SuS algorithm respectively. The aE-SuS algorithm is run 200 times to obtain the mean value, 10 percentile and 90 percentile of the empirical CDF, while a single MCS run with 10^6 samples is carried out for validation.
The average computational cost of aE-SuS is 9,507 calculations of the LSF $g(\cdot)$ and the relative bias of the failure probability is 0.9%, while the coefficient of variation is 0.338. To achieve the same coefficient of variation as aE-SuS, crude MCS needs $1.74 \cdot 10^5$ calculations of the LSF in theory, which is significantly larger than that of aE-SuS. The average CPU time over 200 repetitions of aE-SuS is reported as 682 seconds on a 3.50GHz Intel Xeon E3-1270v3 computer. As a comparison, the CPU time for crude MCS with $1.74 \cdot 10^5$ samples on the same machine is $9.83 \cdot 10^3$ seconds, which is about 14 times larger than the CPU time of aE-SuS.
The standard SuS algorithm is not applicable for this example due to the large jump in the CDF of the LSF.

6. Conclusions

We introduce adaptive effort subset simulation, which enables solving reliability problems with performance functions that follow a discontinuous distribution. Such problems often occur in network reliability assessment because of discrete random variables appearing in the input random vector or due to discontinuities in the function that defines the system performance. The proposed method modifies the adaptive selection of the intermediate domains of the standard SuS and adapts the number of samples and the respective conditional probability throughout the simulation to ensure that there is an adequate number of seeds at each level.
Any MCMC algorithm that enables efficient conditional sampling can be combined with the proposed algorithm. We implement the aCS algorithm in an underlying standard normal space. If the input random variables are

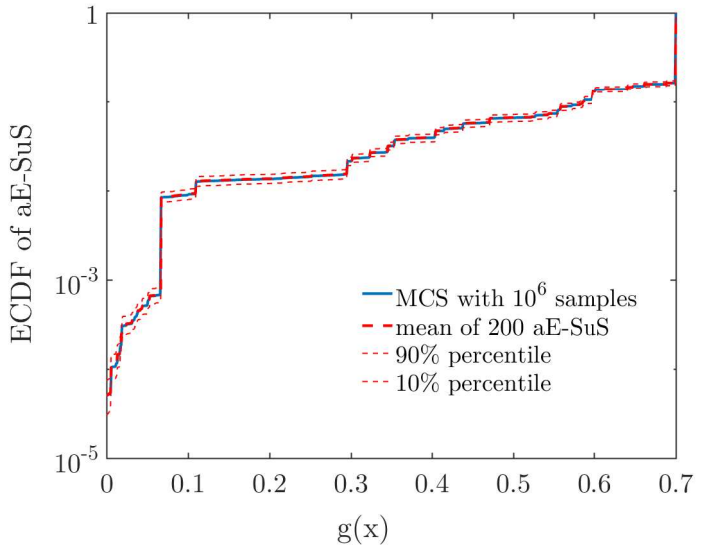


Figure 11: Results obtained by aE-SuS and MCS for IEEE39 network of Example 5.4.

all discrete, we propose a more efficient yet less general independent M-H algorithm, which operates in the original space. The mean acceptance rate of the independent M-H algorithm tends to decrease with increase of the intermediate simulation level with decreasing rate depending on the dimension of the input space. Hence, the algorithm becomes inefficient in estimating small probabilities of high dimensional systems. The acceptance rate of the aCS algorithm is independent of the input dimension, however aCS performs worse than the independent M-H algorithm in moderate dimensional discrete input spaces.

Numerical results demonstrate that the aE-SuS estimator is only slightly biased and has substantially higher efficiency than crude Monte Carlo in problems where standard SuS fails to converge. The computational cost of the aE-SuS algorithm depends highly on the magnitude of the jumps in the distribution of the LSF.

7. Acknowledgment

The first author gratefully acknowledges the financial support of the China Scholarship Council.

Appendix A. Adaptive conditional sampling algorithm

The aCS algorithm presented herein differs from the one of [28], which assumes that $N = \frac{1-p_0}{p_0} N_{seed}$. N is the number of new generated samples and N_{seed} represents the number of seeds. The implementation of the aCS is summarized in Algorithm 4. In the algorithm, l is the intermediate level, and n is the dimension. F_l is the intermediate event. $\mathcal{S}_l$ represents the seeds, and λ_{l-1} is the updated scaling parameter at the $l-1$-th level. It is suggested in [28] to choose λ_0 as 0.6. a^*, N_a, $\{\sigma_{0,d}\}_{d=1}^n$ are respectively the optimal acceptance rate, number of chains to consider for adaption, and the starting values for standard deviation. The suggested values can also be found in [28].

Appendix B. Bound based sampling algorithm

The original bound-based sampling algorithm [12, 8] is proposed for connectivity based problems in multivariate Bernoulli spaces. However, it can be modified to sample from the proposal distribution at level l of the independent M-H algorithm of Section 3.2, $Q_p^{(l)}(\boldsymbol{x}) \propto Q(\boldsymbol{x}|F_0)\mathbb{I}\{\boldsymbol{x} \notin \cup_{j=0,\ldots,l-1}\mathcal{X}_j\}$, if the input random variables follow the multivariate categorical distribution. That is

$$Q(\boldsymbol{x}|F_0) = \prod_{d=1}^{n}\sum_{i=1}^{n_d}\mathbb{I}\{x_d = i\}\theta_{d,i} \tag{B.1}$$

where $\theta_{d,i}$ represents the probability that the d-th component x_d equals value i given all preceding components $x_1, ..., x_{d-1}$. n and n_d represent the number of components and the number of the states of x_d, respectively. For each component d, it holds that $\sum_{i=1}^{n_d}\theta_{d,i} = 1$.

The bound-based sampling algorithm proceeds in a component-wise scheme and is shown in Algorithm 5. Following this algorithm, one generates samples in the space $\{\boldsymbol{x} : \boldsymbol{x} \notin \cup_{j=0,\ldots,l-1}\mathcal{X}_j\}$ with probability proportional to the input distribution $Q(\boldsymbol{x}|F_0)$. A detailed proof can be found in [8].

References

[1] K. M. Zuev, S. Wu, and J. L. Beck, "General network reliability problem and its efficient solution by subset simulation," *Probabilistic Engineering Mechanics*, vol. 40, pp. 25–35, 2015.

Algorithm 4: Adaptive conditional sampling algorithm

Input: N, F_l, $\mathcal{S}^{(l)}$, λ_{l-1}

$N_s \leftarrow |\mathcal{S}^{(l)}|, \lambda \leftarrow \lambda_{l-1}$

Define a^*, N_a, $\{\sigma_{0,d}\}_{d=1}^n$ according to [28]

Randomly sort the seeds $\mathcal{S}^{(l)}$

Randomly choose $\mathrm{mod}(N, N_s)$ elements from the set $\{1, 2, \cdots, N_s\}$, say $\boldsymbol{\chi}$

$\mathcal{P}^{(l)} \leftarrow \emptyset$, $c_1 \leftarrow 0$, $c_2 \leftarrow 0$

for $i = 1, \cdots, N_s$ **do**

  $\sigma_d \leftarrow \min(1, \lambda\sigma_{0,d})$, $\rho_d \leftarrow \sqrt{1-\sigma_d^2}$; $\quad d = 1, \cdots, n$

  if $i \in \boldsymbol{\chi}$ **then**

    $j \leftarrow \lfloor \frac{N}{N_s} \rfloor + 1$

  else

    $j \leftarrow \lfloor \frac{N}{N_s} \rfloor$

  $\boldsymbol{u}_0 \leftarrow$ the i-th seed

  for $k = 1, \cdots, j$ **do**

    for $d = 1, \cdots, n$ **do**

      %% sample the d-th component of candidate $\boldsymbol{v}$

      $v_d \leftarrow \mathcal{N}(\cdot; \rho_d u_{k-1,d}, \sigma_d^2)$

    if $\boldsymbol{v} \in F_l$ **then**

      $\boldsymbol{u}_k \leftarrow \boldsymbol{v}$

      $c_1 \leftarrow c_1 + 1$, $c_2 \leftarrow c_2 + 1$

    else

      $\boldsymbol{u}_k \leftarrow \boldsymbol{u}_{k-1}$

      $c_1 \leftarrow c_1 + 1$

    Add $\boldsymbol{u}_k$ to $\mathcal{P}^{(l)}$

  if *i is a multiple of N_a* **then**

    $\lambda \leftarrow \lambda \exp((\frac{i}{N_a})^{-1/2}[\frac{c_2}{c_1} - a^*])$

    $c_1 \leftarrow 0$, $c_2 \leftarrow 0$

$\lambda_l \leftarrow \lambda$

Output: $\mathcal{P}^{(l)}$, λ_l

Algorithm 5: Bound based sampling algorithm

Input: $Q(\boldsymbol{x}|F_0)$

for $d = 1, \cdots, n$ **do**

  Calculate $\Pr(\boldsymbol{x} \notin \cup_{j=0,\ldots,l-1}\mathcal{X}_j|x_1, \cdots, x_{d-1})$

  for $i = 1, \cdots, n_d$ **do**

    Calculate $\Pr(\boldsymbol{x} \notin \cup_{j=0,\ldots,l-1}\mathcal{X}_j|x_1, \cdots, x_{d-1}, x_d = i)$

    $\theta^*_{d,i} \leftarrow \theta_{d,i}\frac{\Pr(\boldsymbol{x}\notin\cup_{j=0,\ldots,l-1}\mathcal{X}_j|x_1,\cdots,x_{d-1},x_d=i)}{\Pr(\boldsymbol{x}\notin\cup_{j=0,\ldots,l-1}\mathcal{X}_j|x_1,\cdots,x_{d-1})}$

  Sample x_d from the categorical distribution $\mathrm{Cat}(x; \boldsymbol{\theta}^*_d)$

Output: candidate $\boldsymbol{x}$

[2] J. Ching and W.-C. Hsu, "An efficient method for evaluating origin-destination connectivity reliability of real-world lifeline networks," *Computer-Aided Civil and Infrastructure Engineering*, vol. 22, no. 8, pp. 584–596, 2007.

[3] M. Rausand and A. Hoyland, *System reliability theory: models, statistical methods, and applications*. John Wiley & Sons, 2003, vol. 396.

[4] G. S. Fishman, "A Monte Carlo sampling plan for estimating network reliability," *Operations Research*, vol. 34, no. 4, pp. 581–594, 1986.

[5] T. Elperin, I. Gertsbakh, and M. Lomonosov, "Estimation of network reliability using graph evolution models," *IEEE Transactions on Reliability*, vol. 40, no. 5, pp. 572–581, 1991.

[6] H. Cancela and M. El Khadiri, "A recursive variance-reduction algorithm for estimating communication-network reliability," *IEEE Transactions on Reliability*, vol. 44, no. 4, pp. 595–602, 1995.

[7] K.-P. Hui, N. Bean, M. Kraetzl, and D. P. Kroese, "The cross-entropy method for network reliability estimation," *Annals of Operations Research*, vol. 134, no. 1, p. 101, 2005.

[8] P. L'Ecuyer, G. Rubino, S. Saggadi, and B. Tuffin, "Approximate zero-variance importance sampling for static network reliability estimation," *IEEE Transactions on Reliability*, vol. 60, no. 3, pp. 590–604, 2011.

[9] L. Murray, H. Cancela, and G. Rubino, "A splitting algorithm for network reliability estimation," *IIE Transactions*, vol. 45, no. 2, pp. 177–189, 2013.

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

[11] R. Vaisman, D. P. Kroese, and I. B. Gertsbakh, "Splitting sequential Monte Carlo for efficient unreliability estimation of highly reliable networks," *Structural Safety*, vol. 63, pp. 1–10, 2016.

[12] G. Rubino and B. Tuffin, *Rare event simulation using Monte Carlo methods*. John Wiley & Sons, 2009.

[13] R. Paredes, L. Dueñas-Osorio, K. S. Meel, and M. Y. Vardi, "Principled network reliability approximation: A counting-based approach," *Reliability Engineering & System Safety*, vol. 191, p. 106472, 2019.

[14] G. S. Fishman, "The distribution of maximum flow with applications to multistate reliability systems," *Operations Research*, vol. 35, no. 4, pp. 607–618, 1986.

[15] ——, "Monte Carlo estimation of the maximal flow distribution with discrete stochastic arc capacity levels," *Naval Research Logistics (NRL)*, vol. 36, no. 6, pp. 829–849, 1989.

[16] G. S. Fishman and T.-Y. D. Shaw, "Evaluating reliability of stochastic flow networks," *Probability in the Engineering and Informational Sciences*, vol. 3, no. 4, pp. 493–509, 1989.

[17] C. Alexopoulos and G. S. Fishman, "Characterizing stochastic flow networks using the Monte Carlo method," *Networks*, vol. 21, no. 7, pp. 775–798, 1991.

[18] S. Bulteau and M. El Khadiri, "A new importance sampling Monte Carlo method for a flow network reliability problem," *Naval Research Logistics (NRL)*, vol. 49, no. 2, pp. 204–228, 2002.

[19] J. E. Ramirez-Marquez and D. W. Coit, "A Monte-Carlo simulation approach for approximating multi-state two-terminal reliability," *Reliability Engineering & System Safety*, vol. 87, no. 2, pp. 253–264, 2005.

[20] I. Gertsbakh, R. Rubinstein, Y. Shpungin, and R. Vaisman, "Permutational methods for performance analysis of stochastic flow networks," *Probability in the Engineering and Informational Sciences*, vol. 28, no. 1, p. 21, 2014.

[21] P.-C. Chang, "MC-based simulation approach for two-terminal multi-state network reliability evaluation without knowing d-MCs," *Reliability Engineering & System Safety*, vol. 220, p. 108289, 2022.

[22] Y. Zhou, L. Liu, and H. Li, "Reliability estimation and optimisation of multistate flow networks using a conditional Monte Carlo method," *Reliability Engineering & System Safety*, vol. 221, p. 108382, 2022.

[23] S.-K. Au and J. L. Beck, "Estimation of small failure probabilities in high dimensions by subset simulation," *Probabilistic engineering mechanics*, vol. 16, no. 4, pp. 263–277, 2001.

[24] W. Yu, W. Huang, K. Wen, J. Zhang, H. Liu, K. Wang, J. Gong, and C. Qu, "Subset simulation-based reliability analysis of the corroding natural gas pipeline," *Reliability Engineering & System Safety*, vol. 213, p. 107661, 2021.

[25] Z. I. Botev and D. P. Kroese, "Efficient Monte Carlo simulation via the generalized splitting method," *Statistics and Computing*, vol. 22, no. 1, pp. 1–16, 2012.

[26] Z. I. Botev, P. l'Ecuyer, and B. Tuffin, "Reliability estimation for networks with minimal flow demand and random link capacities," *arXiv preprint arXiv:1805.03326*, 2018.

[27] Z. I. Botev and D. P. Kroese, "An efficient algorithm for rare-event probability estimation, combinatorial optimization, and counting," *Methodology and Computing in Applied Probability*, vol. 10, no. 4, pp. 471–505, 2008.

[28] I. Papaioannou, W. Betz, K. Zwirglmaier, and D. Straub, "MCMC algorithms for subset simulation," *Probabilistic Engineering Mechanics*, vol. 41, pp. 89–103, 2015.

[29] I. Papaioannou, S. Geyer, and D. Straub, "Improved cross entropy-based importance sampling with a flexible mixture model," *Reliability Engineering & System Safety*, vol. 191, p. 106564, 2019.

[30] R. Y. Rubinstein and D. P. Kroese, *Simulation and the Monte Carlo method*. John Wiley & Sons, 2016, vol. 10.

[31] M. O. Ball, C. J. Colbourn, and J. S. Provan, "Network reliability," *Handbooks in operations research and management science*, vol. 7, pp. 673–762, 1995.

[32] J.-E. Byun and J. Song, "Generalized matrix-based Bayesian network for multi-state systems," *Reliability Engineering & System Safety*, vol. 211, p. 107468, 2021.

[33] J. He, "An extended recursive decomposition algorithm for dynamic seismic reliability evaluation of lifeline networks with dependent component failures," *Reliability Engineering & System Safety*, vol. 215, p. 107929, 2021.

[34] W.-C. Yeh, "A quick bat for evaluating the reliability of binary-state networks," *Reliability Engineering & System Safety*, vol. 216, p. 107917, 2021.

[35] G. Hardy, C. Lucet, and N. Limnios, "K-terminal network reliability measures with binary decision diagrams," *IEEE Transactions on Reliability*, vol. 56, no. 3, pp. 506–515, 2007.

[36] Z. Wang, M. Broccardo, and J. Song, "Hamiltonian Monte Carlo methods for subset simulation in reliability analysis," *Structural Safety*, vol. 76, pp. 51–67, 2019.

[37] S.-K. Au, "On MCMC algorithm for subset simulation," *Probabilistic Engineering Mechanics*, vol. 43, pp. 117–120, 2016.

[38] C. Andrieu and J. Thoms, "A tutorial on adaptive MCMC," *Statistics and Computing*, vol. 18, no. 4, pp. 343–373, 2008.

[39] W. R. Gilks and P. Wild, "Adaptive rejection sampling for gibbs sampling," *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, vol. 41, no. 2, pp. 337–348, 1992.

[40] A. Scherb, L. Garrè, and D. Straub, "Reliability and component importance in networks subject to spatially distributed hazards followed by cascading failures," *ASCE-ASME Journal of Risk and Uncertainty in Engineering Systems, Part B: Mechanical Engineering*, vol. 3, no. 2, 2017.

[41] H. Rosero-Velásquez and D. Straub, "Representative natural hazard scenarios for risk assessment of spatially distributed," in *The 29th European Safety and Reliability Conference*, 2019, pp. 1–7.

[42] P. Crucitti, V. Latora, and M. Marchiori, "Model for cascading failures in complex networks," *Physical Review E*, vol. 69, no. 4, p. 045104, 2004.