

How to Deal with the Curse of Dimensionality of Likelihood Ratios in Monte Carlo Simulation

Rubinstein R.Y. and P. W. Glynn

Faculty of Industrial Engineering and Management,
Technion, Haifa, Israel;
ierrr01@ie.technion.ac.il

and

Department of Management Science and Engineering,
Stanford University, Stanford
glynn@stanford.edu

December 22, 2008

Abstract

In this work we show how to resolve, at least partially, the curse of dimensionality of likelihood ratios while using importance sampling (IS) to estimate the performance of high-dimensional Monte Carlo simulation problems. The curse of dimensionality, which is better known under the name, the *degeneracy* properties of likelihood ratios, is one of the central topics in Monte Carlo simulation. The current state-of-the-art with IS can be summarized as follows: *do not use IS in high dimensional problems because of the degeneracy properties of likelihood ratios*. We present a simple method, called the *screening* method, which typically allows substantial reduction of the size of the likelihood ratio before applying it. By doing so not only we automatically prevent from the degeneracy of the IS estimators, but at the same time we obtain substantial variance reduction. The main idea behind the screening algorithm is to identify (screen out) the most important parameters of the IS estimator, the so-called *bottleneck* parameter vector, which for typical simulation problems are known to be of low-dimension. As soon as the bottleneck parameter vector is identified, we replace the standard IS estimator with the new low-dimension alternative, where the size of the LR equal to the size of the bottleneck parameter vector. Supportive numerical results are presented.

Keywords. Cross-Entropy, Rare-Event Probability Estimation, Screening, Simulation.

^{0†} This research was supported by the Binational Science Foundation

1 Introduction

The goal of this work is show how to overcome the curse of dimensionality while using likelihood ratios in high-dimensional Monte Carlo simulation problems. The curse of dimensionality, which is know under the name the *degeneracy* properties of likelihood ratios is one of the central topics in Monte Carlo simulation and it is widely discussed in most of the textbooks on Monte Carlo simulation [10]. To prevent degeneracy, several heuristics have been introduced (see, for example, [8]), which did not become popular in the Monte Carlo community because of their poor performance. The current state-of-the art of Monte Carlo simulation can be summarized as follows: *do not use importance sampling (IS) for highly dimensional problems because of the degeneracy properties of likelihood ratios* [10].

In this work we shall show how to overcome this difficulty at least partially. In particular we present a novel method, called the *screening* method, which allows substantial reduction of the size of the likelihood ratios by identifying the most important parameters, called the *bottleneck parameters*. By doing so not only we automatically prevent the degeneracy of IS estimators, but in addition we obtain substantial variance reduction.

We shall deal here mainly with estimating the following rare-event probability

$$\ell(\mathbf{u}, \gamma) = \mathbb{E}_{\mathbf{u}} I_{\{S(\mathbf{X}) \geq \gamma\}}, \quad (1)$$

where

1. $S(\mathbf{X})$ is quite an arbitrary non-negative sample function.
2. $\mathbf{X} = (X_1, \dots, X_n)$ is an n -dimensional random vector of *independent* components.
3. $f(\mathbf{x}, \mathbf{u})$, $\mathbf{u} = (\mathbf{u}_1, \dots, \mathbf{u}_n)$ is a parametric distribution from an exponential family [10]. For simplicity we assume that each marginal pdf $f_r(x)$, $r = 1, \dots, n$ is parameterized by a *single* parameter u_r . Thus $f(\mathbf{x}, \mathbf{u}) = \prod_{r=1}^n f_r(x_r)$.

It will follow from the paper that our results can be extended to dependent random variables and when each marginal pdf $f_r(x)$, $r = 1, \dots, n$ is parameterized by several parameters.

We next assume that $S(\mathbf{X})$ is given such that $\ell(\gamma) \rightarrow 0$ as $\gamma \rightarrow \infty$. Since for large γ the direct estimator of $\ell(\gamma)$ is meaningless, we shall use the following IS estimator [10]

$$\hat{\ell} = \frac{1}{N} \sum_{i=1}^N I_{\{S(\mathbf{X}_i) \geq \gamma\}} W(\mathbf{X}_i, \mathbf{u}, \hat{\mathbf{v}}), \quad (2)$$

where

$$W(\mathbf{X}_i) = \frac{f(\mathbf{X}_i, \mathbf{u})}{g(\mathbf{X}_i)}$$

is the likelihood ratio (LR), $\mathbf{X}_i \sim g(\mathbf{x})$ and $g(\mathbf{x})$ is called the IS pdf.

It is common [10] to use a parametric IS $f(\mathbf{x}, \hat{\mathbf{v}})$ instead of the non-parametric IS pdf $g(\mathbf{x})$, since finding a "good" $g(\mathbf{x})$ is typically very difficult for complex sample function $S(\mathbf{X})$. Here $\hat{\mathbf{v}}$ is an optimal parameter vector obtained from the solution of the conventional programs, like the cross-entropy (CE) and the variance minimization (VM) [9]. In particular, using the CE method $\hat{\mathbf{v}}$ can be derived from the solution of the following stochastic program

$$\max_{\mathbf{v}} \hat{D}(\mathbf{u}, \mathbf{v}) = \max_{\mathbf{v}} \frac{1}{N} \sum_{i=1}^N I_{\{S(\mathbf{X}_i) \geq \gamma\}} W(\mathbf{X}_i, \mathbf{u}, \mathbf{v}) \ln f(\mathbf{X}_i; \mathbf{v}), \quad (3)$$

where

$$W(\mathbf{X}_i, \mathbf{u}, \mathbf{v}) = \frac{f(\mathbf{X}_i, \mathbf{u})}{f(\mathbf{X}_i, \mathbf{v})}$$

and \mathbf{w} is an auxiliary parameter [9]. Similar, the VM counterpart can be written as

$$\max_{\mathbf{v}} \widehat{\mathcal{L}}(\mathbf{u}, \mathbf{v}) = \max_{\mathbf{v}} \frac{1}{N} \sum_{i=1}^N I_{\{S(\mathbf{X}_i) \geq \gamma\}} W(\mathbf{X}_i, \mathbf{u}, \mathbf{w}) W(\mathbf{X}_i, \mathbf{u}, \mathbf{v}). \quad (4)$$

Note that the sample $\mathbf{X}_1, \dots, \mathbf{X}_N$ in (3) - (4) is taken from the pdf $f(\mathbf{x}, \mathbf{w})$. In this work we assume that the components of the random vector \mathbf{X} are independent. Thus, $W(\mathbf{x}, \mathbf{u}, \mathbf{v})$ can be written as

$$W(\mathbf{x}, \mathbf{u}, \mathbf{v}) = \frac{f(\mathbf{x}, \mathbf{u})}{f(\mathbf{x}, \mathbf{v})} = \prod_{k=1}^n \frac{f_k(x_k, \mathbf{u}_k)}{f_k(x_k, \mathbf{v}_k)}.$$

As mentioned, the case where the components of \mathbf{X} are dependent can be treated similar. Note again, that a large vector size n causes the degeneracy of $W(\mathbf{x}, \mathbf{u}, \mathbf{v})$ and thus, poor performance of the estimator of ℓ .

Below we shall use the following facts:

1. We assume that the CE and VM programs (3) and (4) are *convex* with respect to \mathbf{v} . This follows directly from Proposition A.4.1 of [10], provided $f(\mathbf{x}, \mathbf{u})$ is from the exponential family of distributions.
2. Let in addition $\widehat{\mathbf{v}}$ be the optimal solution of the program (3). Then it follows from Proposition A.4.2 of [10] that $\widehat{\mathbf{v}}_k > \mathbf{u}_k$, $k = 1, \dots, n$ component wise, provided $S(\mathbf{x})$ is *monotonically increasing function in each component of the vector \mathbf{x}* , and similar for the program (4).

At this end, recall that for high-dimensional simulation models the programs (3) - (4) are useless, since the LR term W is the product of a large number of marginal likelihoods, and the W terms will cause degeneracy and large variance of the resulting IS estimator $\widehat{\ell}$. On the other hand, importance sampling combined with screening, which involves only a relatively small number of bottleneck elements (and thus a product of a relatively small number of likelihoods) may not only lead to substantial variance reduction, but will produce a stable IS estimator.

The bottleneck phenomenon often occurs when one needs to estimate the probability of a non-typical event in the system, like a rare event probability. For example, if one observes a failure in a system with highly reliable elements, then it is very likely that several elements (typically the less reliable) forming a minimal cut in the model, all fail simultaneously. Another example is the estimation of a buffer overflow probability in a queueing network, that is, the probability that the total number of customers in all queues exceeds some large number. Again, if a buffer overflow occurs, it is quite likely that it has been caused by a buildup in the bottleneck queue, which is the most congested one in the network.

In general, large-dimensional, complex simulation models contain both bottleneck and nonbottleneck parameters. The number of bottleneck parameters is typically smaller than the number of nonbottleneck parameters. Imagine a situation where the size (dimension) of the vector \mathbf{u} is large, say 100, and the number of bottleneck elements is only about 10–15. Then, clearly, an importance sampling estimator based on bottleneck elements alone will not only be much more accurate than its standard importance sampling counterpart involving all 100 likelihood ratios (containing both bottleneck and non-bottleneck ones), but in contrast to the latter will not be degenerated.

The main idea behind the *screening method* is simple and can be outlined as follows:

- The parameter set \mathbf{u} is partitioned as $\mathbf{u} = [\mathbf{u}^{(b)}, \mathbf{u}^{(n)}]$, where $\mathbf{u}^{(b)}$ corresponds to the *bottleneck* part and $\mathbf{u}^{(n)}$ corresponds to the *non-bottleneck* part.
- The IS density $f(\mathbf{x}, \mathbf{v})$ with $\mathbf{v} = [\mathbf{v}^{(b)}, \mathbf{u}^{(n)}]$ changes *only the bottleneck* parameters. Note that the vector $\mathbf{v}^{(b)}$ is typically of much lower size than the vector $\mathbf{u}^{(n)}$.

- A screening algorithm below takes care of identifying the *original* bottleneck parameter vector $\mathbf{u}^{(b)}$.
- Either CE or VM method is applied for estimating the *optimal* bottleneck parameter vector $\mathbf{v}^{(b)}$.

Thus, the screening algorithm contains two-stages. At the first stage we *identify* the set of the elements of the bottleneck parameter vector $\mathbf{v}^{(b)}$, while at the second one we *estimate* the actual values of the components of $\mathbf{v}^{(b)}$ by solving the convex programs (3) - (4).

Let $B \subset \{1, \dots, n\}$ denote the indices of the bottleneck parameters. As soon as B is identified, in the first stage, and the corresponding optimal parameter vector, $\mathbf{v}^{(b)}$ is estimated in the second stage, via $\hat{\mathbf{v}}^{(b)}$, the alternative to $\hat{\ell}$, the so-called *screening* estimator of ℓ , can be written as

$$\hat{\ell}^{(b)} = \frac{1}{N} \sum_{i=1}^N I_{\{S(\mathbf{X}_i) \geq \gamma\}} W^{(b)}(\mathbf{X}_i^{(b)}, \mathbf{u}^{(b)}, \hat{\mathbf{v}}^{(b)}), \quad (5)$$

where

$$W^{(b)}(\mathbf{X}_i^{(b)}, \mathbf{u}^{(b)}, \hat{\mathbf{v}}^{(b)}) = \frac{f^{(b)}(\mathbf{X}_i^{(b)}, \mathbf{u}^{(b)})}{f^{(b)}(\mathbf{X}_i^{(b)}, \hat{\mathbf{v}}^{(b)})},$$

and $\hat{\mathbf{v}}^{(b)}$ denotes the estimate of the bottleneck parameter vector. Note that \mathbf{u} and $\hat{\mathbf{v}}$ can be written as $\mathbf{u} = (\mathbf{u}^{(b)}, \mathbf{u}^{(n)})$ and $\hat{\mathbf{v}} = (\hat{\mathbf{v}}^{(b)}, \hat{\mathbf{v}}^{(n)})$, where $\mathbf{u}^{(n)}$ and $\hat{\mathbf{v}}^{(n)}$ denote the *non-bottleneck* parameter vectors of the original and the estimated reference parameter vectors of the standard IS estimator (2), respectively.

It is crucial to note that in the screening estimator (5) we set automatically $\hat{\mathbf{v}}^{(n)} = \mathbf{u}^{(n)}$.

Consequently, because of the independence of the components, the likelihood ratio term $W(\mathbf{X})$ reduces to a product of $|B|$ quotients of marginal pdfs instead of the product of n such quotients. Note also that as soon as the set B is defined the optimal parameter vector $\hat{\mathbf{v}}^{(b)}$ in (5) can be obtained from the solution of the above standard convex program (3) - (4), provided the pair $\{\mathbf{u}, \mathbf{v}\}$ is replaced by its bottleneck counterpart $\{\mathbf{u}^{(b)}, \mathbf{v}^{(b)}\}$.

The resulting screening estimators of $\mathbf{v}^{(b)}$ obtained by using CE and VM will be called the CE-SCR and VM-SCR estimators, respectively. Observe that since the non-bottleneck parameters are redundant the complexity properties of the estimator $\hat{\ell}^{(b)}$ in (5) can be treated by standard methods [1], [9].

As we shall see from our numerical results below VM-SCR typically outperforms its counterpart CE-SCR in the sense that it has a smaller relative error. In addition, we found that for large n , ($n > 100$), CE under estimates the designed quantity ℓ , while VM-SCR and CE-SCR still perform reasonable well. Our explanation is that for large-size problem CE is affected due to the degeneracy of the LR's.

Most of our simulation results will be carried out for the models composed from bridges, each containing 5 elements. In particular we consider the sample function

$$H(\mathbf{Y}) = \min[\{Y_{11} + \dots + Y_{1m}\}, \dots, \{Y_{n1} + \dots + Y_{nm}\}], \quad (6)$$

where each unit Y_{ij} , $i = 1, \dots, n$; $j = 1, \dots, m$ presents a bridge defined as

$$Y_{ij} = \min \left\{ \begin{array}{ll} \{X_{ij1} + X_{ij4}\}, & \{X_{ij2} + X_{ij5}\}, \\ \{X_{ij1} + X_{ij3} + X_{ij5}\}, & \{X_{ij2} + X_{ij3} + X_{ij4}\} \end{array} \right\}$$

and the sample function

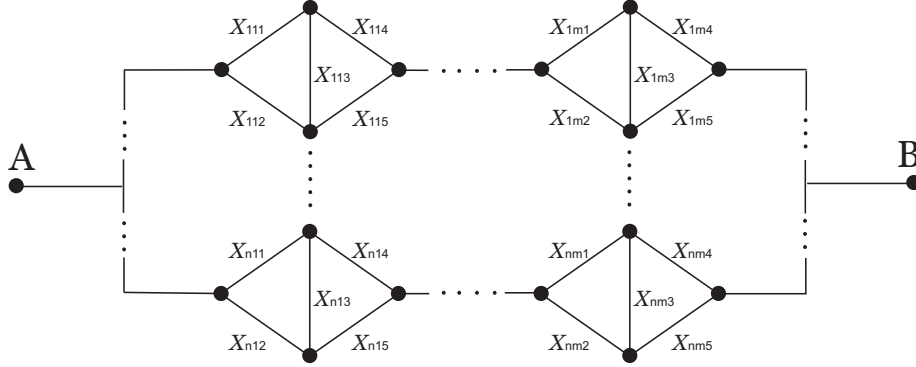
$$H(\mathbf{Y}) = \max[\min\{Y_{11}, \dots, Y_{1m}\}, \dots, \min\{Y_{n1}, \dots, Y_{nm}\}], \quad (7)$$

where

$$Y_{ij} = \max \left\{ \begin{array}{ll} \min\{X_{ij1}, X_{ij4}\}, & \min\{X_{ij2}, X_{ij5}\}, \\ \min\{X_{ij1}, X_{ij3}, X_{ij5}\}, & \min\{X_{ij2}, X_{ij3}, X_{ij4}\} \end{array} \right\}.$$

Figure 1 corresponds to formula (7) and presents a network of size $n \times m$ combined of bridges, each containing 5 elements.

Figure 1: A network combined of bridges



Note that for Bernoulli random variables the sample function $S(\mathbf{X})$ in (7), can be viewed as structure function of a reliability model [7].

We consider separately the screening algorithm for general distributions and for the Bernoulli ones. We shall also present case studies with our algorithms to estimate rare-event probabilities associated with the performance of a network taken from Fishman [7].

In sections 2 and 3 we present our main algorithms and their numerical results corresponding to general pdf's and the Bernoulli pdf, respectively. In Section 4 some conclusions are derived.

2 The Two-Stage Screening Algorithm for General Distributions

Here we present the two-stage screening algorithm for general distributions. Recall that at the first stage we identify the set of the bottleneck parameter vector $\mathbf{v}^{(b)}$, while at the second one we estimate the actual values of the components of $\mathbf{v}^{(b)}$ using the convex programs (3) - (4).

The main idea of the first stage of our screening algorithm is to identify the bottleneck parameters without *involving the LR term* W . One may wonder how this can be possible if estimation of the rare-event probability $\ell(\mathbf{u}) = \mathbb{E}_{\mathbf{u}} I_{\{S(\mathbf{X}) \geq \gamma\}}$ by itself is essentially based on IS and, thus on LR's. The trick is to execute the first stage (the screening part) by replacing the *quantile* γ with some γ_0 such that $\ell(\mathbf{u}, \gamma_0) = \mathbb{E}_{\mathbf{u}} I_{\{S(\mathbf{X}) \geq \gamma_0\}}$ is *not* a rare-event probability, which is assumed to be fixed. Denote $\ell(\mathbf{u}, \gamma_0)$ by ρ . Typically we set $0.01 \leq \rho \leq 0.1$.

To estimate γ_0 from simulation, we start by generating a sample $\mathbf{X}_1, \dots, \mathbf{X}_N$ from $f(\mathbf{x}, \mathbf{u})$. Denote by $\tilde{\mathcal{X}}_0 = \{\tilde{\mathbf{X}}_1, \dots, \tilde{\mathbf{X}}_{N_0}\}$ the largest subset of the population $\{\mathbf{X}_1, \dots, \mathbf{X}_N\}$, corresponding to the ordered statistics values of $S(\mathbf{X}_1), \dots, S(\mathbf{X}_N)$ for which $S(\mathbf{X}_i) \geq \gamma_0$. Note that γ_0 is the $(1 - \rho)$ sample quantile of the equation $\mathbb{E}_{\mathbf{u}} I_{\{S(\mathbf{X}) \geq \gamma_0\}} = \rho$. In short, for fixed ρ , $\tilde{\mathcal{X}}_0 = \{\tilde{\mathbf{X}}_1, \dots, \tilde{\mathbf{X}}_{N_0}\}$ corresponds to the

largest subset of the population $\{\mathbf{X}_1, \dots, \mathbf{X}_N\}$, for which $S(\mathbf{X}_i) \geq \gamma_0$. For more details see [9].

As soon as γ_0 is estimated from simulation, the execution of the first stage reduces to finding the optimal parameter vector $\hat{\mathbf{v}}$ based on CE and VM. This can be executed by solving the programs (3)-(4) with the advantage that there is no need for the LR terms $W(\mathbf{X}, \mathbf{u}, \mathbf{w})$. For concreteness we rewrite only (3) by omitting W 's and by replacing γ with $\hat{\gamma}_0$. We have

$$\max_{\mathbf{v}} \hat{D}(\mathbf{v}) = \max_{\mathbf{v}} \frac{1}{N} \sum_{i=1}^N I_{\{S(\mathbf{X}_i) \geq \hat{\gamma}_0\}} \ln f(\mathbf{X}_i; \mathbf{v}). \quad (8)$$

Note that the sample $\mathbf{X}_1, \dots, \mathbf{X}_N$ in (8) is taken from the original pdf $f(\mathbf{x}, \mathbf{u})$ and $\hat{\gamma}_0$ presents the $(1 - \rho)$ -sample quantile of the performance $S(\mathbf{X})$. If not stated otherwise we refer below to CE program (8).

Recall that for the CE method the parameter vector $\hat{\mathbf{v}}$ (and $\hat{\mathbf{v}}^{(b)}$) can often be found analytically, in particular when the sampling distribution comes from an exponential family parameterized by the mean. In contrast to CE, finding $\hat{\mathbf{v}}$ (and $\hat{\mathbf{v}}^{(b)}$) in VM typically involves a numerical procedure. We shall present only the detailed algorithm based on CE-SCR. Its VM-SCR counterpart is similar.

Consider the first stage of our algorithm, that is identification of the set B of the bottleneck parameter vector $\mathbf{v}^{(b)}$. This stage is associated with a simple classification procedure which divides the n -dimensional vector $\hat{\mathbf{v}}$ into two parts, namely as $\hat{\mathbf{v}} = (\hat{\mathbf{v}}^{(b)}, \hat{\mathbf{v}}^{(n)})$ and such that we set in advance (component wise) $\hat{\mathbf{v}}^{(n)} = \mathbf{u}^{(n)}$, while $\hat{\mathbf{v}}^{(b)} \neq \mathbf{u}^{(b)}$ (component wise). It can be readily shown in analogy to Proposition A.4.2 of [10] that if each component of the random vector \mathbf{X} is from the exponential family parameterized by the mean and if $S(\mathbf{x})$ is a monotonically increasing function in each component of \mathbf{x} on the interval $[0, \infty)$, then each element of $\mathbf{v}^{(b)}$ is at least as large as the corresponding one $\mathbf{u}^{(b)}$.

2.1 The Screening Method for $\ell = \mathbb{E}_{\mathbf{u}}[H(\mathbf{X})]$

We present below two separate screening algorithms: one for estimating the performance

$$\ell = \ell(\mathbf{u}) = \mathbb{E}_{\mathbf{u}}[H(\mathbf{X})],$$

where $H(\mathbf{X})$ is assumed to be a quite arbitrary sample function and another one for an indicator performance function $I_{\{S(\mathbf{X}) \geq \gamma\}}$. Here \mathbf{X} is a n -dimensional random vector with a quite arbitrary parametric distribution $f(\mathbf{x}; \mathbf{u})$. In the latter case we assume that γ is very large, so ℓ is a rare-event probability, say $\ell \leq 10^{-6}$. In the former case the screening method (applying IS to a small subset of $\mathbf{v}^{(b)}$) might be helpful if $H(\mathbf{X})$ is a high-dimensional noisy function. In such case screening might lead to substantial variance reduction. Another example is estimating systems unreliability with highly reliable components considered in Section 3.

We rewrite for completeness the stochastic program (8) with $I_{\{S(\mathbf{X}) \geq \gamma\}}$ replaced by $H(\mathbf{X})$ as

$$\max_{\mathbf{v}} \hat{D}(\mathbf{v}) = \max_{\mathbf{v}} \frac{1}{N} \sum_{i=1}^N H(\mathbf{X}_i) \ln f(\mathbf{X}_i; \mathbf{v}). \quad (9)$$

Algorithm 2.1 (CE-SCR Screening Algorithm for Estimating $\mathbb{E}_{\mathbf{u}}H(\mathbf{X})$)

1. Initialize the set of bottleneck elements to $B_0 = \{1, \dots, n\}$. Set $t = 1$.
2. Generate a sample $\mathbf{X}_1, \dots, \mathbf{X}_N$ from $f(\mathbf{x}; \mathbf{u})$ and deliver the CE solution of the stochastic program (9). Denote the solution by $\hat{\mathbf{v}}_t = (\hat{v}_{t1}, \dots, \hat{v}_{tn})$. Note that $\hat{\mathbf{v}}_t$ is an n -dimensional parameter vector.

- Calculate the relative perturbation for each element \widehat{v}_{ti} , $i = 1, \dots, n$ as

$$\delta_{ti} = \frac{\widehat{v}_{ti} - u_i}{u_i}. \quad (10)$$

- If $\delta_{ti} < \delta$, where δ is some threshold value, say $\delta = 0.1$ (note that negative δ_{ti} automatically satisfies $\delta_{ti} < \delta$), set $\widehat{v}_{ti} = u_i$, that is, identify the i -th element of the vector \mathbf{v} as a *non-bottleneck* parameter. Otherwise identify it as a *bottleneck* one. Let B_t be the set of bottleneck elements at iteration t .
- Repeat steps 2–4. several times, say d times, increasing each time t by 1, and updating the set B_t until convergence.
- Apply the standard CE program (9) to estimate the optimal parameter vector $\mathbf{v}^{(b)}$, with $B = B_d$.
- Apply smoothing, that is calculate

$$\widehat{\mathbf{p}}_t^{(b)} = \alpha \widetilde{\mathbf{p}}_t^{(b)} + (1 - \alpha) \widehat{\mathbf{p}}_{t-1}^{(b)}, \quad (11)$$

where α , ($0 < \alpha < 1$) is called the *smoothing parameter*.

- Deliver (5) (with $I_{\{S(\mathbf{x}) \geq \gamma\}}$ replaced by $H(\mathbf{X})$) as the resulting estimator of the rare-event probability ℓ .

Note that

- In all our numerical studies we observed that the sequence of sets B_t , $t = 1, \dots, d$ is non-decreasing and it converges quite fast (2-4 iterations) to the true bottleneck set B_d .
- The smoothing step (11) can be viewed as a fine tuning one. It is commonly used in iterative simulation-based algorithms [9].

It is important to note the following:

- As mentioned, under the present assumptions (independent components, each from a one-parameter exponential family parameterized by the mean, and $H(\mathbf{x})$ monotonically increasing in each component), the components of the \mathbf{v}^* are at least as large as the corresponding elements of \mathbf{u} . Taking this into account, Algorithm 2.1 always identifies all elements i corresponding to $\delta_i < 0$ as non-bottleneck ones.
- Recall that Steps 2–4 are purposely performed d times. This allows one to better determine the nonbottleneck parameters, since it is likely that they will fluctuate around their nominal value u_i and therefore δ_i will become negative or very small in one of the replications.
- The advantage of Algorithm 2.1 compared to its gradient counterpart is that identification of the bottleneck elements in the former is based on the relative perturbations δ_i (see (10)) with respect to the *known* original parameter values u_i , while in the latter it is based on the absolute value of the gradient itself. It is not difficult to see that the former classifier, the so-called $\widehat{\mathbf{v}}$ -based classifier, is more natural to use than the gradient-based one. In addition, we found numerically that it identifies more accurately the actual bottleneck size.

Numerical Results

We now present numerical studies with Algorithm 2.1 for the model (1), (6) consisting of $n \times m$ bridges arranged in a grid, provided the operator min in (6) is replaced by max.

In our numerical results we assume that the components X_{ijk} of the random vector \mathbf{X} are independent each distributed $\text{Weib}(\alpha, u)$, that is, X_{ijk} has the density

$$f(x; \alpha, \beta) = \alpha\beta^{-\alpha} x^{\alpha-1} e^{-(x/\beta)^\alpha},$$

with $\beta = u^{-1/\alpha}$ and $u = u_{ijk}$. Note [9] that a Weibull random variable can be generated using the transformation $\beta Z^{1/\alpha}$, where $Z \sim \text{exp}(1)$. We also assume that only u is controllable, while α is fixed and equals 0.2. We purposely selected some elements of \mathbf{u} to be bottleneck ones, and set $\delta = 0.1$.

Table 1 presents the performance of Algorithm 2.1 for the 1×1 (single bridge) model (6), with $\ell = \mathbb{E}_{\mathbf{u}} H(\mathbf{Y})$, where the sample function $H(\mathbf{Y}) = Y_{11}$ and

$$Y_{ij} = \min \left\{ \begin{array}{ll} \{X_{111} + X_{114}\}, & \{X_{112} + X_{115}\}, \\ \{X_{111} + X_{113} + X_{115}\}, & \{X_{112} + X_{113} + X_{114}\} \end{array} \right\}$$

Here the $u_{111} = 1$ and $u_{112} = 1$ in $f(\mathbf{x}, \mathbf{u})$, $bu = (u_{111}, \dots, u_{115})$ are chosen to be the bottleneck parameters, whereas the remaining (non-bottleneck) ones we set equal to 2. The notations in Table 1 are as follows:

1. “Mean, max and min $\widehat{\ell}$ ” denote the sample mean, maximum and minimum and minimal values of the 10 estimates of $\widehat{\ell}$.
2. “RE” denotes the sample relative error for $\widehat{\ell}$, averaged over the 10 runs.
3. Mean T - denotes the average number of iterations based on 10 runs.
4. “CPU” denotes the average CPU time in seconds based on 10 runs.
5. N and N_1 denote the sample size, while updating the parameter vector \mathbf{v} and estimating ℓ , respectively.
6. “CMC” denotes the crude Monte Carlo.

Table 1: Performance of Algorithm 2.1 for the single bridge model with samples $N = N_1 = 500$

Method		CMC	CE	VM	CE-SCR	VM-SCR
$\widehat{\ell}$	Mean	4.052	3.970	3.734	3.894	3.829
	Max	8.102	4.327	4.201	4.345	4.132
	Min	1.505	3.380	3.395	3.520	3.278
RE		0.5188	0.0704	0.0776	0.0755	0.0674
Mean T		0.0	3.0	3.0	3.0	3.0
Mean CPU		0.00	0.04	0.21	0.05	0.13

It follows from the results of Table 1 that for this relatively small model both CE and VM perform similarly to their screening counterparts and they outperform the CMC. We will further see that as complexity of the models increases, VM-SCR outperforms its alternatives and in particular it outperforms CE-SCR. Note also that for this model both methods CE and VM identified correctly the 2 bottleneck parameters.

Table 2 presents a typical dynamics of identifying the 2 bottleneck parameters at the first stage of Algorithm 2.1 for the above single bridge model having a total of 5 parameters. In Table 2, t denotes the replication number at the first stage, while the rest of it comprises 0’s and 1’s. Here 0 and 1 mean that an appropriate parameter is identified as non-bottleneck and bottleneck one, respectively. One can see that after 2 replications we have 4 bottleneck parameters left, after 6 replications there are 3 bottleneck parameters left, and after 7 replication the process stabilizes identifying correctly the true 2 bottleneck parameters.

Table 2: Typical dynamics for identifying the bottleneck parameters at the first stage of Algorithm 2.1 for the bridge model

t	u_1	u_2	u_3	u_4	u_5
0	1	1	1	1	1
1	1	1	0	1	1
2	1	1	0	1	1
3	1	1	0	1	0
4	1	1	0	1	0
5	1	1	0	1	0
6	1	1	0	1	0
7	1	1	0	0	0

Table 3 presents a typical evolution of the sequence $\{\widehat{v}_t\}$ in the single bridge model for the VM and VM-SCR methods at the second stage of Algorithm 2.1.

Table 3: Typical evolution of the sequence $\{\widehat{v}_t\}$ for the VM and VM-SCR methods

VM						VM-SCR					
t	\widehat{v}_1	\widehat{v}_2	\widehat{v}_3	\widehat{v}_4	\widehat{v}_5	t	\widehat{v}_1	\widehat{v}_2	\widehat{v}_3	\widehat{v}_4	\widehat{v}_5
	1.000	1.000	2.000	2.000	2.000		1.000	1.000	2	2	2
1	0.537	0.545	2.174	2.107	1.615	1	0.555	0.599	2	2	2
2	0.346	0.349	2.071	1.961	1.914	2	0.375	0.402	2	2	2
3	0.306	0.314	1.990	1.999	1.882	3	0.315	0.322	2	2	2

One can clearly see that the bottleneck parameters decrease about 3 times after the third iteration, while the non-bottleneck ones fluctuate about their nominal value $u = 2$.

Table 4 presents performance of Algorithm 2.1 for the 3x10 model with 6 bottlenecks corresponding to the elements $u_{111}, u_{112}, u_{211}, u_{212}, u_{311}, u_{312}$. We set $u_{111} = u_{112} = u_{211} = u_{212} = u_{311} = u_{312} = 1$, while the remaining (non-bottlenecks) values are set equal 2. Note again that in this case both CE and VM found exactly the true 6 bottlenecks.

Table 4: Performance of Algorithm 2.1 for the 3x10 model with 6 bottleneck elements and sample size $N = N_1 = 1000$

Method		CMC	CE	VM	CE-SCR	VM-SCR
$\widehat{\ell}$	Mean	16.16	16.11	14.84	16.12	15.67
	Max	22.65	26.85	16.59	18.72	17.20
	Min	11.13	7.007	12.59	14.63	14.80
RE		0.2036	0.3427	0.0745	0.0743	0.0489
Mean T		0.0	3.0	3.0	3.0	3.0
Mean CPU		0.00	0.49	68.36	0.73	27.54

It follows from the result of Table 4 that without screening even the naive Monte Carlo outperforms the standard CE. However, using screening one obtains substantial improvement of CE. Finally, VM-SCR outperforms all its alternatives.

2.2 The Screening Method for Rare Events

Here we show how the screening method works for estimating rare-event probabilities of the form

$$\ell = \mathbb{P}_{\mathbf{u}}(S(\mathbf{X}) \geq \gamma) = \mathbb{E}_{\mathbf{u}}[I_{\{S(\mathbf{X}) \geq \gamma\}}],$$

where as before we assume that the components of \mathbf{X} are independent, that each component is distributed according to a one-dimensional exponential family parameterized by the mean, and that $S(\mathbf{x})$ (and hence $H(\mathbf{x}) = I_{\{S(\mathbf{x}) \geq \gamma\}}$) is monotonically increasing in each component of \mathbf{x} . In particular, we shall present a modification of the two-stage Algorithm 2.1.

As in Algorithm 2.1 the main idea of the first stage of the modified algorithm is to identify the bottleneck parameters *without involving the LR terms* W . Recall that the trick is to execute the first stage (the screening part) by replacing γ with some γ_0 and such that $\ell(\mathbf{u}, \gamma_0) = \mathbb{E}_{\mathbf{u}} I_{\{S(\mathbf{X}) \geq \gamma_0\}}$ is *not* a rare-event probability, say $10^{-2} \leq \ell(\mathbf{u}, \gamma_0) \leq 10^{-1}$. As soon as γ_0 is determined, the execution of the first stage is similar to the one in Algorithm 2.1. It reduces to finding the estimator $\widehat{\mathbf{v}} = \widehat{\mathbf{v}}(\gamma_0)$ of the optimal parameter vector $\mathbf{v}^* = \mathbf{v}^*(\gamma_0)$, obtained from (8), where γ is replaced by γ_0 . Note that (8) does not contain the LR term $W(\mathbf{X}, \mathbf{u}, \mathbf{w})$. It is important to note again that since we assume that $\widehat{\ell}(\mathbf{u})$ is monotone non-decreasing function in each component of the vector \mathbf{u} and the associated CE and VM programs are convex, we can classify the bottleneck and not bottleneck parameters according to the relative perturbation

$$\delta_{tr} = \frac{\widehat{v}_{tr} - u_r}{u_r}, \quad r = 1, \dots, k; \quad t = 1, \dots, d,$$

which is the core of the screening algorithm.

Algorithm 2.2 (CE-SCR Screening Algorithm for Rare-Events)

1. The same as in Algorithm 2.1.
2. Generate a sample $\mathbf{X}_1, \dots, \mathbf{X}_N$ from $f(\mathbf{x}; \mathbf{u})$ and compute $\widehat{\gamma}_0$, the $(1-\rho)$ -sample quantile of the sample performances $S(\mathbf{X}_i)$.
3. Generate a different sample $\mathbf{X}_1, \dots, \mathbf{X}_N$ from $f(\mathbf{x}; \mathbf{u})$ and deliver the CE solution of the stochastic program (8), with $\gamma = \widehat{\gamma}_0$. Denote the solution by $\widehat{\mathbf{v}}_t = (\widehat{v}_{t1}, \dots, \widehat{v}_{tn})$. Note that $\widehat{\mathbf{v}}_t$ is a parameter vector from $f(\mathbf{x}; \widehat{\mathbf{v}}_t)$.
- 4.- 8. The same as Steps 3.-7. in Algorithm 2.1.
9. Deliver (5) as the resulting estimator of the rare-event probability ℓ .

Numerical Results

We next present numerical studies with Algorithm 2.2 for the model (6) composed from bridges and the random variables X_{ijk} are distributed $\exp(u_{ijk})$, which is the same as $\text{Weib}(\alpha, u_{ijk})$, but with $\alpha = 1$.

As before we purposely selected in advance that elements of our model to be bottlenecks. Recall that for the exponential pdf $u \exp(-ux)$, $x \geq 0$ the u values corresponding to the bottleneck elements should be smaller than for the non-bottleneck ones.

Table 5 presents the performance of Algorithm 2.2 for the 2x2 model with 8 bottlenecks using $\delta = 0.1$, $\gamma = 6$ and the sample sizes $N = 50,000$ and $N_1 = 500,000$. In particular we set the 6 bottlenecks, which are u_{111} , u_{112} , u_{121} , u_{122} , u_{211} , u_{212} , u_{221} , u_{222} equal 1, while the remaining 12 elements we set equal to 4. Note that for this simple model both methods CE and VM identified correctly (at the first stage) the 8 bottleneck parameters.

Table 5: Performance of Algorithm 2.2 for the 2x2 model. We set $\delta = 0.1$, $\gamma = 6$, $N = 50,000$, $N_1 = 500,000$

Method		CE	VM	CE-SCR	VM-SCR
$\hat{\ell}$	Mean	2.918E-08	2.956E-08	2.881E-08	2.814E-08
	Max	3.933E-08	3.686E-08	3.563E-08	3.289E-08
	Min	2.464E-08	2.648E-08	2.540E-08	2.447E-08
RE		0.16609	0.10193	0.10861	0.07682
Mean T		8.0	9.1	8.0	10.4
Mean CPU		6.03	9.31	6.56	9.12

It follows from the results of Table 5 that for this relatively small model both CE and VM perform similarly to their screening counterparts. We will further see that as the complexity of the model increases VM-SCR outperforms its 3 alternatives and in particular its CE-SCR counterpart.

Table 6 represents a typical dynamics of identifying the bottleneck parameters at the first stage of Algorithm 2.2 for the above 2x2 model with 20 parameters, 8 of which are bottlenecks. Similar to Table 2 in Table 6 the 0's and the 1's mean that an appropriate parameter u is identified as non-bottleneck and bottleneck one, respectively and t denotes the replication number at the first stage of the algorithm. It is readily seen that after the first replication we have 13 bottleneck parameters, after the second one - 11 bottleneck parameters, and after the third replication the process stabilizes delivering the 8 true bottleneck parameters.

Table 6: A typical dynamics for identifying the bottleneck parameters at the first stage of Algorithm 2.2

r	u_{111}	u_{112}	u_{113}	u_{114}	u_{115}	u_{121}	u_{122}	u_{123}	u_{124}	u_{125}
0	1	1	1	1	1	1	1	1	1	1
1	1	1	1	0	0	1	1	1	0	0
2	1	1	0	0	0	1	1	0	0	0
3	1	1	0	0	0	1	1	0	0	0
4	1	1	0	0	0	1	1	0	0	0
5	1	1	0	0	0	1	1	0	0	0
r	u_{211}	u_{212}	u_{213}	u_{214}	u_{215}	u_{221}	u_{222}	u_{223}	u_{224}	u_{225}
0	1	1	1	1	1	1	1	1	1	1
1	1	1	1	0	0	1	1	0	1	1
2	1	1	1	0	0	1	1	0	1	1
3	1	1	0	0	0	1	1	0	0	0
4	1	1	0	0	0	1	1	0	0	0
5	1	1	0	0	0	1	1	0	0	0

Table 7 presets typical evolution of the sequence $\{(\hat{\gamma}_t, \hat{v}_t)\}$ for the elements of the first bridge of the above 2x2 model for the VM and VM-SCR methods.

Table 7: Typical evolution of the sequence $\{(\widehat{\gamma}_t, \widehat{\mathbf{v}}_t)\}$ for the VM and VM-SCR methods

VM						
t	$\widehat{\gamma}_t$	\bar{v}_{111}	\bar{v}_{112}	\bar{v}_{113}	\bar{v}_{114}	\bar{v}_{115}
0		1.000	1.000	4.000	4.000	4.000
1	2.331	0.759	0.771	3.944	3.719	3.839
2	2.993	0.635	0.613	3.940	3.681	3.734
3	3.559	0.524	0.517	4.060	3.297	3.608
4	4.176	0.443	0.415	3.370	3.353	3.909
5	4.863	0.334	0.332	3.689	3.965	4.250
6	5.574	0.378	0.365	3.827	3.167	4.188
7	6.000	0.357	0.358	3.881	4.235	4.929
8	6.000	0.285	0.271	4.011	2.982	4.194
9	6.000	0.287	0.301	3.249	2.879	3.409
VM-SCR						
t	$\widehat{\gamma}_t$	\bar{v}_{111}	\bar{v}_{112}	\bar{v}_{113}	\bar{v}_{114}	\bar{v}_{115}
0		1.000	1.000	4.000	4.000	4.000
1	2.357	0.760	0.771	4.000	4.000	4.000
2	2.958	0.638	0.605	4.000	4.000	4.000
3	3.541	0.506	0.491	4.000	4.000	4.000
4	4.055	0.486	0.447	4.000	4.000	4.000
5	4.532	0.402	0.371	4.000	4.000	4.000
6	5.010	0.348	0.317	4.000	4.000	4.000
7	5.551	0.375	0.347	4.000	4.000	4.000
8	6.000	0.285	0.298	4.000	4.000	4.000
9	6.000	0.288	0.254	4.000	4.000	4.000
10	6.000	0.278	0.277	4.000	4.000	4.000

It follows from the result of Table 7 that the bottleneck elements decrease more than 3 times, while the non-bottleneck ones fluctuate about their nominal values equal 4.

We proceed next with some larger models. For these models we purposely chose the following 6 parameters u_{111} , u_{112} , u_{211} , u_{212} , u_{311} , u_{312} as bottleneck ones, while the remaining as non-bottleneck ones.

Table 8 presents the performance of Algorithm 2.2 for the 3x5 model, where we set $u_{111} = u_{211} = u_{311} = 1.5$, $u_{112} = u_{212} = u_{312} = 1$, while the remaining (non-bottleneck) ones we set equal to 2. We also set $\delta = 0.05$, $\gamma = 6$ and the sample size $N = N_1 = 500,000$.

In this case CE and VM identified at the first stage 56 and 44 bottleneck parameters, respectively out of the total of 75 parameters. The reason for that is, the bottleneck elements are not so evident as in the previous 2x2 model.

Table 8: Performance of Algorithm 2.2 for the 3x5 model. We set $\delta = 0.05$, $\gamma = 6$, $\rho=0.001$, $N = 500,000$ and $N_1 = 500,000$

Method		CE	VM	CE-SCR	VM-SCR
$\widehat{\ell}$	Mean	1.062E-07	8.728E-08	8.011E-08	8.061E-08
	Max	2.629E-07	1.711E-07	9.372E-08	9.874E-08
	Min	7.428E-08	3.502E-08	6.363E-08	6.283E-08
RE		0.53069	0.49580	0.15226	0.14253
Mean T		6.0	6.9	6.0	7.3
Mean CPU		148.49	231.92	146.73	214.09

It follows that in this case both CE-SCR and VM-SCR have similar relative error (accuracy) and they both outperform their CE and VM counterparts, respectively.

Table 9 presents performance of Algorithm 2.2 for the 3x10 model with the above 6 bottlenecks, which were set to 1, that is we set $u_{111} = u_{211} = u_{311} = u_{112} = u_{212} = u_{312} = 1$, while the remaining ones were set to 4. We also set $\delta = 0.1$, $\gamma = 6$, $N = N_1 = 400,000$. In this case both CE and VM found the true 6 bottlenecks.

Table 9: Performance of Algorithm 2.2 for the 3x10 model with 6 bottlenecks. We set $\delta = 0.1$, $\gamma = 6$, $N = 400,000$, $N_1 = 400,000$

Method		CE	VM	CE-SCR	VM-SCR
$\widehat{\ell}$	Mean	2.440E-08	5.343E-08	5.288E-08	5.175E-08
	Max	5.825E-08	7.180E-08	8.384E-08	6.932E-08
	Min	4.148E-15	2.763E-08	2.746E-08	4.326E-08
RE		1.05557	0.28452	0.32857	0.15232
Mean T		6.0	6.0	7.4	8.9
Mean CPU		247.15	482.36	303.92	531.27

Note that VM-SCR is the most accurate among its three alternatives, and that CE *under estimates* ℓ . Thus, for this relatively large model with 150 elements CE (without screening) is affected by the degeneracy of the LR.

We next consider a network from [7] depicted in Figure 2. In particular, we consider estimation of the rare-event probability $\ell(\mathbf{u}) = \mathbf{E}_{\mathbf{u}} I_{\{S(\mathbf{X}) \geq \gamma\}}$ with the performance $S(\mathbf{X})$ being the shortest path from node 1 to 20, $\mathbf{X} \sim \exp(\mathbf{u})$, where \mathbf{u} and γ are fixed in advance. Note that we use Dijkstra's algorithm for calculating the shortest path in the network. Note also that using full enumeration we found that the total number of feasible path in the network equals 830. Note also that in this case the bottleneck elements are not available in advance.

Figure 2: A network

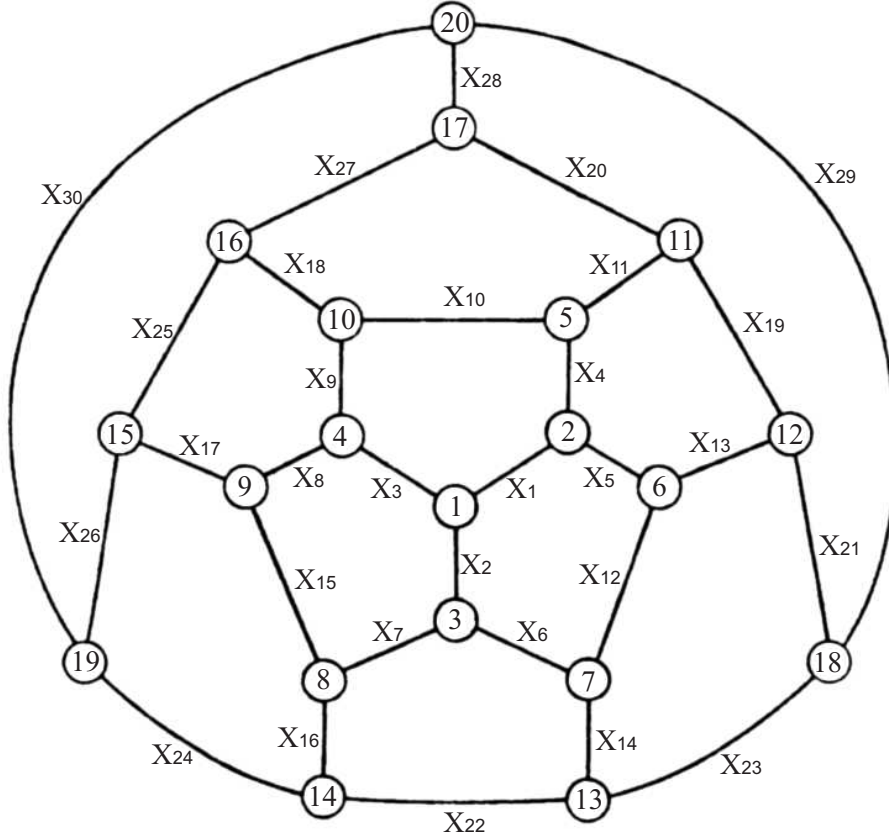


Table 10 presents the performance of Algorithm 2.2 for the model in Fig.2 with $\gamma = 10$, $N = 50,000$ and $N_1 = 100,000$. We set all 20 parameters equal to 1 and selected $\rho = 0.01$ and $\delta = 0.05$. For this model CE and VM identified at the first stage of Algorithm 2.2, 26 and 24 bottleneck elements (out of the total of 30 elements in the network), respectively.

Table 10: Performance of Algorithm 2.2 for the model in Fig.2 with all initial parameters equal 1, $\delta = 0.05$. $\gamma = 10$, $N = 50,000$, $N_1 = 100,000$

Method		CE	VM	CE-SCR	VM-SCR
$\hat{\ell}$	Mean	1.233E-08	1.234E-08	1.134E-08	1.341E-08
	Max	2.076E-08	1.629E-08	1.669E-08	2.142E-08
	Min	8.640E-09	8.997E-09	9.118E-09	9.529E-09
RE		0.33293	0.20606	0.20532	0.29083
Mean T		7.0	8.1	7.0	8.7
Mean CPU		36.32	45.44	36.45	48.07

It follows from above that all 4 approaches perform similarly. The reason while screening does not help much here is that in this case the number of bottlenecks found

is close to the total number of elements. According to our nomenclature this model can be viewed as one with not evident parameters.

While updating the parameter vector $\hat{\mathbf{v}}^{(b)}$ at the second stage of Algorithm 2.2 with the above model we found that 6 elements of $\hat{\mathbf{v}}^{(b)}$ corresponding to the parameters $u_1, u_2, u_3, u_{28}, u_{29}, u_{30}$ have change the most. Taking this into consideration we can easily modify some of the initial unity parameters of the network in such a way that the above network could be viewed as one with evident bottlenecks. In particular, one could keep, for example, all 6 initial bottleneck parameters $u_1, u_2, u_3, u_{28}, u_{29}, u_{30}$ equal to 1, while increasing the remaining (non-bottleneck) ones.

Table 11 presents data for such a case. In particular it presents data similar to Table 10 but with Weib($\alpha, u^{-1/\alpha}$) pdf instead of $\exp(u)$ pdf. As before we assume that only u is controllable, while all α 's are equal to 1/4. In addition, we set $u_1 = u_2 = u_3 = u_{28} = u_{29} = u_{30} = 1$, while the remaining 24 ones we set equal to 4. For this model we found that both CE and VM correctly identified the above 6 bottleneck elements.

Table 11: Performance of Algorithm 2.2 for the model in Fig.2 with $\delta = 0.1$, $\gamma = 2000$, $N = 100,000$, $N_1 = 300,000$

Method		CE	VM	CE-SCR	VM-SCR
$\hat{\ell}$	Mean	7.415E-09	4.013E-09	4.045E-09	4.331E-09
	Max	4.628E-08	6.138E-09	4.991E-09	5.387E-09
	Min	9.744E-11	2.784E-09	3.065E-09	3.257E-09
RE		1.85189	0.24994	0.15102	0.19062
Mean T		9.6	12.6	9.4	15.8
Mean CPU		109.85	260.44	109.80	215.28

As expected, for this model (with more evident bottlenecks), both screening versions, CE-SCR and VM-SCR outperform their non-screening ones.

3 Screening for Bernoulli Models

Note that for Bernoulli random variables and in particular, while estimating systems *unreliability* with highly reliable components one needs to modify the first stage of Algorithm 2.2. The main reason is that since most of the components of the Bernoulli vector \mathbf{u} are close to 1 (the system consists of highly reliable components) it is useless to run Algorithm 2.2 by generating a sample *directly* from the original pdf $f(\mathbf{x}, \mathbf{u}) = \text{Bern}(\mathbf{u})$. Consider, for example, the toy example of estimating $\ell = \mathbb{E}_{\mathbf{u}}(X)$, where $X \sim \text{Bern}(u)$ and $u = 0.999$. By taking a sample of size $N = 1,000$ from such $\text{Bern}(0.999)$, we would get in average only one 0, while the rest would be 1's.

To overcome this difficulty we shall introduce an auxiliary Bernoulli parameter vector \mathbf{p}_0 (for Bernoulli pdf we shall use the notation \mathbf{p} instead of \mathbf{v} , as in the previous section), which will play the role similar to γ_0 in $\ell(\gamma_0) = \mathbb{E}_{\mathbf{u}} I_{\{S(\mathbf{x}) \geq \gamma_0\}}$. This can be done, for example, as follows. Set, say to 0.85 the values of \mathbf{p}_0 corresponding to \mathbf{u} , which are greater than 0.85; set the remaining elements of \mathbf{p}_0 equal to the corresponding elements of \mathbf{u} . Thus, the elements of the original vector \mathbf{u} , which are greater than 0.85 are set in \mathbf{p}_0 automatically to 0.85, while the remaining ones are adopted by \mathbf{p}_0 without any change. It is readily seen that by doing so $\ell(\mathbf{p}_0) = \mathbb{E}_{\mathbf{u}} I_{\{S(\mathbf{x}) \geq \mathbf{p}_0\}}$ will be not a rare event probability any more, and thus, while estimating $\ell(\mathbf{p}_0)$ we shall generate enough 0's and 1's.

Algorithm 3.1 (CE-SCR Algorithm for the Bernoulli Distribution)

1. Select \mathbf{p}_0 as follows: set to 0.85 the values of \mathbf{p}_0 corresponding to \mathbf{u} , which are greater than 0.85; set the remaining elements of \mathbf{p}_0 equal to the corresponding elements of \mathbf{u} .
2. The same as Step 1 in Algorithm 2.1.
3. Generate a sample $\mathbf{X}_1, \dots, \mathbf{X}_N$ from the Bernoulli pdf $f(\mathbf{x}, \hat{\mathbf{p}}_{t-1}) = \text{Ber}(\hat{\mathbf{p}}_{t-1})$ and deliver the solution of the stochastic program (see (8))

$$\max_{\mathbf{p}} \hat{D}(\mathbf{p}) = \max_{\mathbf{p}} \frac{1}{N} \sum_{i=1}^N I_{\{S(\mathbf{X}_i)=0\}} W^{(b)}(\mathbf{X}_i^{(b)}; \mathbf{u}^{(b)}, \hat{\mathbf{p}}_{t-1}^{(b)}) \ln f(\mathbf{X}_i^{(b)}; \mathbf{p}). \quad (12)$$

Denote the solution by $\hat{\mathbf{p}}_t = (\hat{p}_{t1}, \dots, \hat{p}_{tk})$. Note also that in contrast to (8), $W(\mathbf{X}_i, \mathbf{u}, \mathbf{p}_0) = 1$ *only* for the elements of \mathbf{p}_0 which were set *equal* to \mathbf{u} .

4. Calculate the relative perturbation for each element \hat{p}_{tr} , $r = 1, \dots, k$ as

$$\delta_{tr} = \frac{u_r - \hat{p}_{tr}}{u_r}. \quad (13)$$

- 5-8. The same as Steps 4.-7. in Algorithm 2.1.
5. Deliver (see also (5))

$$\hat{\ell}^{(b)} = \frac{1}{N} \sum_{i=1}^N I_{\{S(\mathbf{X}_i)=0\}} W^{(b)}(\mathbf{X}_i^{(b)}; \mathbf{u}^{(b)}, \hat{\mathbf{p}}^{(b)}),$$

as the resulting estimator of the rare-event probability ℓ .

Numerical Results

We start with the numerical results corresponding to the reliability system for the model in Figure 1 when the sample function $S(\mathbf{X})$ is defined in (7). Similar to the tables in Section 2 we consider here models with more evident and less evident bottlenecks. Clearly that in the first case the reference Bernoulli parameters in the bottleneck elements must be chosen *smaller* than the remaining ones, while in the second case the bottleneck elements must be chosen quite *close* to the remaining ones. As in Section 2

1. We set $\alpha = 0.7$ and performed 10 independent runs.
2. We automatically set all Bernoulli reference parameters p corresponding to $\delta > 0$ back to their nominal values u , that is we view them as non-bottleneck ones.

Tables 12 presents the performance of Algorithm 3.1, for the 2x2 model, where we set $N = 10,000$, $N_1 = 50,000$, $\rho = 0.01$, $\delta = 0.1$; $u_{ijk} = 0.97$ for $k = 1, 2$ and all $(i, j) = 1, 2$, and we set $u_{ijk} = 0.99$ for $k = 3, 4, 5$ and all $(i, j) = 1, 2$. We view such model as the one with evident bottleneck elements. Clearly, in this case there are 8 bottleneck elements, corresponding to $u_{ijk} = 0.99$. All 8 bottleneck elements were identified correctly by both CE and VM methods.

Table 12: Performance of Algorithm 3.1 for the 2x2 model with evident bottle-necks. We set $N = 10,000$, $N_1 = 50,000$, $\rho = 0.01$, $\delta = 0.1$

Method		CMC	CE	VM	CE-SCR	VM-SCR
$\hat{\ell}$	Mean	2.000E-6	2.798E-6	3.264E-6	3.244E-6	3.202E-6
	Max	2.000E-5	3.572E-6	3.431E-6	3.339E-6	3.328E-6
	Min	0.000E+0	1.601E-6	3.080E-6	3.150E-6	3.089E-6
RE		3.16228	0.29473	0.02927	0.01960	0.02511
Mean T		0.0	8.0	8.0	8.0	8.0
Mean CPU		0.00	1.54	10.03	1.33	5.87

Table 13 presents a typical evolution of the sequence $\{\hat{p}_t\}$ in the first bridge of the above 2x2 model for the VM and VM-SCR methods at the second stage of Algorithm 3.1.

Table 13: Typical evolution of sequence $\{\hat{p}_t\}$

VM					
	u_{111}	u_{112}	u_{113}	u_{114}	u_{115}
	0.970	0.970	0.990	0.990	0.990
t	p_{111}	p_{112}	p_{113}	p_{114}	p_{115}
1	0.820	0.820	0.840	0.840	0.840
2	0.557	0.561	0.968	0.894	0.884
3	0.551	0.509	0.977	0.941	0.939
4	0.528	0.520	0.970	0.941	0.962
5	0.505	0.503	0.987	0.974	0.988
6	0.496	0.496	0.940	0.983	0.986
7	0.495	0.495	0.977	0.990	0.985
8	0.491	0.493	0.985	0.993	0.987
VM-SCR					
	u_{111}	u_{112}	u_{113}	u_{114}	u_{115}
	0.970	0.970	0.990	0.990	0.990
t	p_{111}	p_{112}	p_{113}	p_{114}	p_{115}
0	0.760	0.760	0.990	0.990	0.990
2	0.492	0.489	0.990	0.990	0.990
3	0.617	0.616	0.990	0.990	0.990
4	0.528	0.528	0.990	0.990	0.990
5	0.503	0.506	0.990	0.990	0.990
6	0.593	0.591	0.990	0.990	0.990
7	0.532	0.536	0.990	0.990	0.990
8	0.506	0.508	0.990	0.990	0.990

One can clearly see that the bottleneck parameters corresponding to $u = 0.99$ decrease about 2 times already after the third iteration, while the non-bottleneck ones fluctuate about their nominal value $u = 0.99$.

We next present simulation results for larger models, namely for 3x5 and 3x10 ones, where we set $u_{i11} = u_{i12} = 0.95$ for $i = 1, 2, 3$, while the rest we set equal to 0.99. We also set $\rho = 0.01$ and $\delta = 0.1$. Clearly, both models have the same 6

bottlenecks, which were correctly by the CE and VM methods.

Table 14 presents the performance of the Algorithm 3.1 for the 3x5 model with $N = 50,000$ and $N_1 = 250,000$ samples.

Table 14: Performance of Algorithm 3.1 for the 3x5 model with $N = 50,000$ and $N_1 = 250,000$ samples

Method		CE	VM	CE-SCR	VM-SCR
$\hat{\ell}$	Mean	2.983E-08	3.450E-08	2.257E-08	2.146E-08
	Max	5.147E-08	6.816E-08	3.327E-08	2.206E-08
	Min	2.129E-08	2.643E-08	1.862E-08	2.106E-08
RE		0.31565	0.35840	0.17860	0.01513
Mean T		8.0	8.0	8.0	8.0
Mean CPU		25.35	371.14	28.64	182.04

Table 15 presents similar data for 3x10 model. We set $N = 100,000$, $N_1 = 300,000$.

Table 15: Performance of Algorithm 3.1 for the 3x10 model with samples $N = 100,000$, $N_1 = 300,000$

Method		CE	VM	CE-SCR	VM-SCR
$\hat{\ell}$	Mean	4.583E-08	5.909E-08	4.096E-08	4.699E-08
	Max	1.412E-07	1.079E-07	4.545E-08	5.671E-08
	Min	1.156E-09	4.423E-08	3.025E-08	3.689E-08
RE		0.82762	0.33868	0.11162	0.16357
Mean T		8.0	8.0	8.0	8.0
Mean CPU		99.80	1699.48	106.48	1065.15

It follows that both VM-SCR and CE-SCR perform accurately and they both outperform substantially (in the RE sense) their standard VM and CE counterparts. We finally consider again the model from [7] depicted in Figure 2.

Table 16 presents performance of Algorithm 3.1 with all equal initial parameters u , which were chosen $u = 0.999$. We set $N = 10,000$, $N_1 = 50,000$, $\alpha = 0.7$, $\rho = 0.01$ and $\delta = 0.01$. While updating the parameter vector $\hat{\mathbf{p}}^{(b)}$ at the second stage of Algorithm 3.1 we found that the 6 elements of $\hat{\mathbf{p}}^{(b)}$ corresponding to the parameters $u_1, u_2, u_3, u_{28}, u_{29}, u_{30}$, which changed most and which were correctly identified by the VM method. Note that CE identified only 3 bottlenecks. Because of this CE-SCR performs poorly as Table 16 shows.

Table 16: Performance of Algorithm 3.1 with equal initial parameters $u = 0.999$, $N = 10,000$, $\rho = 0.01$, $N_1 = 50,000$, $\alpha = 0.7$ and $\delta = 0.01$

Method		CMC	CE	VM	CE-SCR	VM-SCR
$\hat{\ell}$	Mean	0.000E+0	2.232E-6	2.073E-6	1.000E-6	2.048E-6
	Max	0.000E+0	3.858E-6	2.184E-6	1.000E-6	2.164E-6
	Min	0.000E+0	1.008E-6	1.996E-6	1.000E-6	1.968E-6
RE		NaN	0.45903	0.03720	0.00005	0.04300
Mean T		0.0	8.0	8.0	8.0	8.0
Mean CPU		0.00	8.18	27.96	9.17	19.67

Taking this into consideration that 6 parameters $u_1, u_2, u_3, u_{28}, u_{29}, u_{30}$ changed most we can easily modify the initial parameter vector \mathbf{u} of the network in such a way that the above network could be viewed as one with evident bottlenecks. In particular, one can, for example, increase all 6 initial bottleneck parameters $u_1, u_2, u_3, u_{28}, u_{29}, u_{30}$, while keeping the remaining (non-bottleneck) ones equal as before 0.999.

Table 17 presents data for such a case. In particular it presents data similar to Table 16 where we set the 6 bottlenecks from the previous experiment equal to 0.97, while we keep the remaining 24 parameters equal to 0.999. In this case CE identified 13 bottlenecks, while VM identified correctly 22 ones. The results of Table 17 are self explanatory.

Table 17: Performance of Algorithm 3.1 with less evident bottlenecks. $N = 10,000$, $\rho = 0.01$, $N_1 = 50,000$, $\alpha = 0.7$, $\delta = 0.01$

Method		CMC	CE	VM	CE-SCR	VM-SCR
$\hat{\ell}$	Mean	6.000E-5	4.353E-5	5.436E-5	5.436E-5	5.442E-5
	Max	1.000E-4	5.527E-5	5.593E-5	5.758E-5	5.731E-5
	Min	2.000E-5	2.700E-5	5.233E-5	5.225E-5	5.204E-5
RE		0.49690	0.32695	0.02323	0.03037	0.03092
Mean T		0.0	8.0	8.0	8.0	8.0
Mean CPU		0.00	11.93	16.18	9.88	12.93

4 Conclusions

In this work we showed how to overcome the curse of dimensionality of likelihood ratios in high-dimensional Monte Carlo simulation problems, caused by their degeneracy properties [10]. In particular we presented a method, called the *screening* method, which allows substantial reduction of the size of the likelihood ratios by identifying the most important parameters, called the *bottleneck parameters*. By doing so not only we automatically prevent the degeneracy of IS estimators, but in addition we obtain substantial variance reduction.

Our extensive numerical studies clearly indicate that

1. For models with quite evident bottlenecks the two-stage screening algorithms are quite efficient for both CE-SCR and VM-SCR versions, provided $0.05 \leq \delta \leq 0.1$. The VM-SCR version is typically the most efficient one and its relative efficiency increases with the size of the model.

2. For models with less evident bottlenecks the VM-SCR version is still typically the most accurate one, provided $\delta \approx 0.01$. The effect of screening is, however, not as dramatic as for models with evident bottlenecks.
3. As the size of the models increases the efficiency of VM-SCR relative to its 3 counterparts, CE, VM and CE-SCR increases. For large-size models, like $n \geq 150$ we found that
 - (a) Both CE and VM performs poorly because of the degeneracy of the LR's. Typically the degeneracy of VM occurs for larger size models than for CE.
 - (b) Both CE-SCR and VM-SCR perform nicely.
 - (c) Although CE-SCR is faster VM-SCR is typically more accurate.

References

- [1] Asmussen S. and Glynn P. *Stochastic Simulation*, Springer, 2007.
- [2] Ben-Tal A., D.E. Brown and R.L. Smith. "Relative Entropy and the Convergence of the Posterior and Empirical Distributions under Incomplete and Conflicting Information". Manuscript, University of Michigan, 1988.
- [3] Ben-Tal A. and M. Taboulle. "Penalty Functions and Duality in Stochastic Programming via ϕ -Divergence Functionals." *Mathematics of Operations Research*, Vol 12, No2, pp 224-240, 1987.
- [4] Blanchet J. "Importance Sampling and efficient Counting of 0-1 Contingency Tables. Manuscript, Harvard University, 2006.
- [5] Cover T.M. and Thomas J.A., *Elements of Information Theory*, John Wiley & Sons, inc, 1991.
- [6] Kapur J.N. and H.K. Kesavan, *Entropy Optimization with Applications*, Academic Press, Inc., 1992.
- [7] G. Fishman *Monte Carlo*. Springer, 1995.
- [8] Liu J.S. *Monte Carlo Strategies in Scientific Computing*. Springer, 2001.
- [9] Rubinstein R.Y. and Kroese D.P., *The Cross-Entropy Method: a Unified Approach to Combinatorial Optimization, Monte-Carlo Simulation and Machine Learning*, Springer, 2004.
- [10] R. Y. Rubinstein and D. P. Kroese, *Simulation and the Monte Carlo Method; Second Edition*, Wiley, 2007.