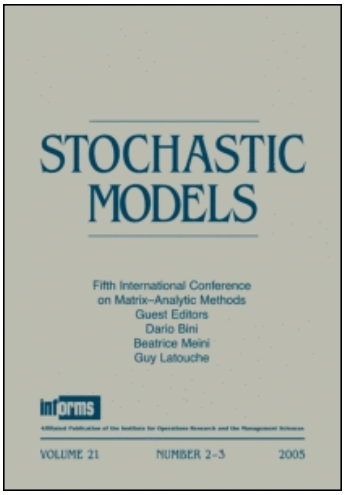

Stochastic Models
Publication details, including instructions for authors and subscription information:
http://www.informaworld.com/smpp/title~content=t713597301

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

Reuven Y. Rubinstein[a]; Peter W. Glynn[b]
[a] Faculty of Industrial Engineering and Management, Technion, Haifa, Israel [b] Department of Management Science and Engineering, Stanford University, Stanford, Connecticut, USA

To cite this Article Rubinstein, Reuven Y. and Glynn, Peter W.(2009) 'How to Deal with the Curse of Dimensionality of Likelihood Ratios in Monte Carlo Simulation', Stochastic Models, 25: 4, 547 — 568
To link to this Article: DOI: 10.1080/15326340903291248
URL: http://dx.doi.org/10.1080/15326340903291248

Stochastic Models, 25:547–568, 2009

ISSN: 1532-6349 print/1532-4214 online
DOI: 10.1080/15326340903291248

Taylor & Francis
Taylor & Francis Group

HOW TO DEAL WITH THE CURSE OF DIMENSIONALITY OF LIKELIHOOD RATIOS IN MONTE CARLO SIMULATION

Reuven Y. Rubinstein[1] and Peter W. Glynn[2]

[1]*Faculty of Industrial Engineering and Management, Technion, Haifa, Israel*
[2]*Department of Management Science and Engineering, Stanford University, Stanford, Connecticut, USA*

□ *In this work we show how to resolve, at least partially, the curse of dimensionality of likelihood ratios (LRs) while using importance sampling (IS) to estimate the performance of high-dimensional Monte Carlo simulation problems. The curse of dimensionality, which is better known as* degeneracy *properties of LR, 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 LR before applying it. By doing so we not only automatically prevent the degeneracy of the IS estimators, but obtain substantial variance reduction at the same time. 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 equals the size of the bottleneck parameter vector. Supportive numerical results are presented.*

Keywords Cross-entropy; Rare-event probability estimation; Screening; Simulation.

Mathematics Subject Classification 60G99.

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 known by the name of degeneracy properties of likelihood ratios, is one of the

Received June 2007; Accepted June 2009
Address correspondence to Reuven Y. Rubinstein, Department of Industrial Engineering and Management, Technion University, Haifa, Israel; E-mail: ierrr01@ie.technion.ac.il

central topics in Monte Carlo simulation and is widely discussed in most textbooks on Monte Carlo simulation[5]. To prevent degeneracy, several heuristics have been introduced (see, for example, Ref.[3]), 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[5].

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 we not only automatically prevent the degeneracy of IS estimators, but in addition we obtain substantial variance reduction.

We shall deal here with estimating the following two expected values:

$$\ell(\boldsymbol{u}) = \mathbb{E}_{\boldsymbol{u}} S(\boldsymbol{X}) \tag{1}$$

and

$$\ell(\boldsymbol{u}, \gamma) = \mathbb{E}_{\boldsymbol{u}} I_{\{S(\boldsymbol{X}) \geq \gamma\}}, \tag{2}$$

where

1. $S(\boldsymbol{X})$ is quite an arbitrary non-negative sample function.
2. $\boldsymbol{X} = (X_1, \ldots, X_n)$ is an n-dimensional random vector of independent components.
3. γ is a fixed parameter.
4. $f(\boldsymbol{x}, \boldsymbol{u}), \boldsymbol{u} = (\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n)$ is a parametric distribution from an exponential family[5]. For simplicity, we assume that each marginal pdf $f_r(x)$, $r = 1, \ldots, n$ is parameterized by a single parameter u_r. Thus $f(\boldsymbol{x}, \boldsymbol{u}) = \prod_{r=1}^{n} f_r(x_r)$.

Note that in the former case (1), the screening method might be helpful if $S(\boldsymbol{X})$ is a high-dimensional noisy function. In such a case, screening might lead to substantial variance reduction. Another example is estimating systems unreliability with highly reliable components considered in section 3. In the latter case (2), we assume that γ is very large, say ℓ is a rare-event probability, say $\ell \leq 10^{-6}$.

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

We next assume that $S(\boldsymbol{X})$ is given such that $\ell(\gamma) \to 0$ as $\gamma \to \infty$. Since for large γ the direct estimator of $\ell(\gamma)$ is meaningless ($\ell(\boldsymbol{u}, \gamma)$ is a rare-event probability), we shall use the following IS estimator[5]:

$$\hat{\ell} = \frac{1}{N} \sum_{i=1}^{N} I_{\{S(\boldsymbol{X}_i) \geq \gamma\}} W(\boldsymbol{X}_i, \boldsymbol{u}), \tag{3}$$

where

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

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

It is common[5] to use a parametric IS $f(\boldsymbol{x}, \hat{\boldsymbol{v}})$ instead of the nonparametric IS pdf $g(\boldsymbol{x})$, since finding a "good" $g(\boldsymbol{x})$ is typically very difficult for complex sample function $S(\boldsymbol{X})$. Here $\hat{\boldsymbol{v}}$ is an optimal parameter vector obtained from the solution of the conventional programs, like the cross-entropy (CE) and the variance minimization (VM)[4]. In particular, using the CE method $\hat{\boldsymbol{v}}$ can be derived from the solution of the following stochastic program[4]:

$$\max_{\boldsymbol{v}} \widehat{D}(\boldsymbol{u}, \boldsymbol{v}) = \max_{\boldsymbol{v}} \frac{1}{N} \sum_{i=1}^{N} I_{\{S(\boldsymbol{X}_i) \geq \gamma\}} W(\boldsymbol{X}_i, \boldsymbol{u}, \boldsymbol{w}) \ln f(\boldsymbol{X}_i; \boldsymbol{v}), \tag{4}$$

where

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

and $\boldsymbol{w}$ is an auxiliary parameter[4]. Similarly, the VM counterpart can be written as

$$\min_{\boldsymbol{v}} \widehat{\mathscr{D}}(\boldsymbol{u}, \boldsymbol{v}) = \min_{\boldsymbol{v}} \frac{1}{N} \sum_{i=1}^{N} I_{\{S(\boldsymbol{X}_i) \geq \gamma\}} W(\boldsymbol{X}_i, \boldsymbol{u}, \boldsymbol{w}) W(\boldsymbol{X}_i, \boldsymbol{u}, \boldsymbol{v}). \tag{5}$$

Note that the sample $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N$ in (4) and (5) is taken from the pdf $f(\boldsymbol{x}, \boldsymbol{w})$. In this work we assume that the components of the random vector $\boldsymbol{X}$ are independent. Thus, $W(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{v})$ can be written as

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

As mentioned, the case where the components of $\boldsymbol{X}$ are dependent can be treated similar. Note again, that a large vector size n causes the degeneracy of $W(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{v})$ 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 (4) and (5) are convex with respect to $\boldsymbol{v}$. This follows directly from Proposition A.4.1 of Ref.[5], provided $f(\boldsymbol{x}, \boldsymbol{u})$ is from the exponential family of distributions.
2. Let in addition $\hat{\boldsymbol{v}}$ be the optimal solution of the program (4). Then it follows from Proposition A.4.2 of Ref.[5] that $\hat{\boldsymbol{v}}_k > \boldsymbol{u}_k$, $k = 1, \ldots, n$ componentwise, provided $S(\boldsymbol{x})$ is monotonically increasing function in each component of the vector $\boldsymbol{x}$, and similar for the program (5).

At this end, recall that for high-dimensional simulation models, the programs (4) and (5) 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 $\hat{\ell}$. 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 nontypical 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 with light-tailed distributions, where we are interested in 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 build-up 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 $\boldsymbol{u}$ 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 nonbottleneck 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 $\boldsymbol{u}$ is partitioned as $\boldsymbol{u} = [\boldsymbol{u}^{(b)}, \boldsymbol{u}^{(n)}]$, where $\boldsymbol{u}^{(b)}$ corresponds to the bottleneck part and $\boldsymbol{u}^{(n)}$ corresponds to the nonbottleneck part.

- The IS density $f(\boldsymbol{x}, \boldsymbol{v})$ with $\boldsymbol{v} = [\boldsymbol{v}^{(b)}, \boldsymbol{u}^{(n)}]$ changes only the bottleneck parameters. Note that the vector $\boldsymbol{v}^{(b)}$ is typically of much lower size than the vector $\boldsymbol{u}^{(n)}$.
- A screening algorithm below takes care of identifying the original bottleneck parameter vector $\boldsymbol{u}^{(b)}$.
- Either CE or VM method is applied for estimating the optimal bottleneck parameter vector $\boldsymbol{v}^{(b)}$.

Thus, the screening algorithm contains two stages. At the first stage, we identify the set of the elements of the bottleneck parameter vector $\boldsymbol{v}^{(b)}$, while at the second one we estimate the actual values of the components of $\boldsymbol{v}^{(b)}$ by solving the convex programs (4) and (5).

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

$$\hat{\ell}^{(b)} = \frac{1}{N} \sum_{i=1}^{N} I_{\{S(\boldsymbol{X}_i) \geq \gamma\}} W^{(b)}\left(\boldsymbol{X}_i^{(b)}, \boldsymbol{u}^{(b)}, \hat{\boldsymbol{v}}^{(b)}\right), \tag{6}$$

where

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

$\hat{\boldsymbol{v}}^{(b)}$ denotes the estimate of the bottleneck parameter vector $\boldsymbol{v}^{(b)}$ and $f^{(b)}(\boldsymbol{X}_i^{(b)}, \hat{\boldsymbol{v}}^{(b)})$ corresponds to the bottleneck part of the joint pdf $f(\boldsymbol{x}, \boldsymbol{u})$.

Consequently, because of the independence of the components, the likelihood ratio term $W(\boldsymbol{X})$ 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 $\hat{\boldsymbol{v}}^{(b)}$ in (6) can be obtained from the solution of the above standard convex programs (4) and (5), provided the pair $\{\boldsymbol{u}, \boldsymbol{v}\}$ is replaced by its bottleneck counterpart $\{\boldsymbol{u}^{(b)}, \boldsymbol{v}^{(b)}\}$.

The resulting screening estimators of $\boldsymbol{v}^{(b)}$ obtained by using CE and VM will be called the CE-SCR (screening) and VM-SCR estimators, respectively. Observe that since the nonbottleneck parameters are redundant the complexity properties of the estimator $\hat{\ell}^{(b)}$ in (6) can be treated by standard methods[1,4].

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

reasonably well. Our explanation is that for large-size problems CE is affected due to the degeneracy of the LRs.

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

$$S(\boldsymbol{Y}) = \max[\{Y_{11} + \cdots + Y_{1m}\}, \ldots, \{Y_{n1} + \cdots + Y_{nm}\}], \tag{7}$$

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

$$Y_{ij} = \min \begin{Bmatrix} \{X_{ij1} + X_{ij4}\}, & \{X_{ij2} + X_{ij5}\}, \\ \{X_{ij1} + X_{ij3} + X_{ij5}\}, & \{X_{ij2} + X_{ij3} + X_{ij4}\} \end{Bmatrix}$$

and the sample function

$$S(\boldsymbol{Y}) = \max[\min\{Y_{11}, \ldots, Y_{1m}\}, \ldots, \min\{Y_{n1}, \ldots, Y_{nm}\}], \tag{8}$$

where

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

Figure 1 corresponds to formula (8) and presents a network of size $n \times m$ combined of bridges, each containing five elements.

Note that for Bernoulli random variables, the sample function $S(\boldsymbol{X})$ in (8) can be viewed as a structure function of a reliability model[2].

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[2].

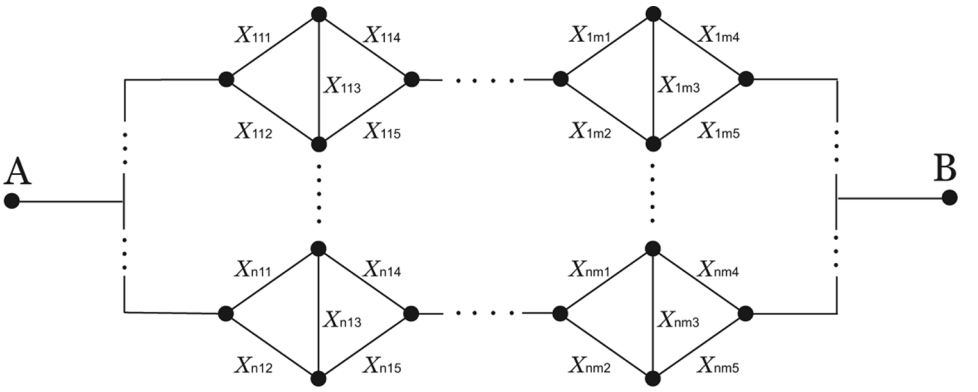

FIGURE 1 A network combined of bridges.

In sections 2.2 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 SCREENING METHOD

We present below two separate screening algorithms: one for estimating ℓ in (1) and another for estimating ℓ in (2).

Recall that for the CE method, the parameter vector $\hat{\boldsymbol{v}}$ (and $\hat{\boldsymbol{v}}^{(b)}$) 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 $\hat{\boldsymbol{v}}$ (and $\hat{\boldsymbol{v}}^{(b)}$) in VM typically involves a numerical procedure. We shall present only the detailed algorithm based on CE-SCR. Its VM-SCR counterpart is similar.

2.1. The Screening Algorithm for $\ell = \mathbb{E}_{\boldsymbol{u}}[S(\boldsymbol{X})]$

We shall use here the stochastic program (4) without likelihood ratio terms W's and with $I_{\{S(\boldsymbol{X})\geq\gamma\}}$ replaced by $S(\boldsymbol{X})$, that is the program

$$\max_{\boldsymbol{v}} \widehat{D}(\boldsymbol{v}) = \max_{\boldsymbol{v}} \frac{1}{N} \sum_{i=1}^{N} S(\boldsymbol{X}_i) \ln f(\boldsymbol{X}_i; \boldsymbol{v}). \tag{9}$$

Algorithm 2.1 (CE-SCR Screening Algorithm for Estimating $\mathbb{E}_{\boldsymbol{u}} S(\boldsymbol{X})$)**.**

1. Initialize the set of bottleneck elements to $B_0 = \{1, \ldots, n\}$. Set $t = 1$.
2. Generate a sample $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N$ from $f(\boldsymbol{x}; \boldsymbol{u})$ and deliver the CE solution of the stochastic program (9). Denote the solution by $\hat{\boldsymbol{v}}_t = (\hat{v}_{t1}, \ldots, \hat{v}_{tn})$.
3. Calculate the relative perturbation for each element $\hat{v}_{ti}$, $i = 1, \ldots, n$ as

$$\delta_{ti} = \frac{\hat{v}_{ti} - u_i}{u_i}. \tag{10}$$

4. 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 $\hat{v}_{ti} = u_i$, that is, identify the ith element of the vector $\boldsymbol{v}$ as a nonbottleneck parameter. Otherwise identify it as a bottleneck one. Let B_t be the set of bottleneck elements at iteration t.
5. Repeat steps 2–4 several times, say d times, increasing each time t by 1, and updating the set B_t until convergence.
6. Apply the standard CE program (9) to estimate the optimal parameter vector $\boldsymbol{v}^{(b)}$, with $B = B_d$. Denote the estimator of $\boldsymbol{v}^{(b)}$ as $\tilde{\boldsymbol{v}}^{(b)}$.

7. Apply smoothing[4], that is calculate

$$\hat{\boldsymbol{v}}_t^{(b)} = \alpha \tilde{\boldsymbol{v}}_t^{(b)} + (1-\alpha)\hat{\boldsymbol{v}}_{t-1}^{(b)}, \tag{11}$$

where α, $(0 < \alpha < 1)$ is called the smoothing parameter.
8. Deliver (6) (with $I_{\{S(\boldsymbol{X})\geq\gamma\}}$ replaced by $S(\boldsymbol{X})$) as the resulting estimator of the performance measure ℓ.

It can be readily shown in analogy to Proposition A.4.2 of Ref.[5] that if each component of the random vector $\boldsymbol{X}$ is from the exponential family parameterized by the mean and if $S(\boldsymbol{x})$ is a monotonically increasing function in each component of $\boldsymbol{x}$ on the interval $[0, \infty)$, then each element of $\boldsymbol{v}^{(b)}$ is at least as large as the corresponding one $\boldsymbol{u}^{(b)}$.

Note also that

- In all our numerical studies, we observed that the sequence of sets $B_t, t = 1, \ldots, d$ is a nondecreasing one.
- The smoothing step (11) can be viewed as a fine tuning one. It is commonly used in iterative simulation-based algorithms[4].

It is important to note the following:

1. As mentioned, under the present assumptions (independent components, each from a one-parameter exponential family parameterized by the mean, and $S(\boldsymbol{x})$ monotonically increasing in each component), the components of $\hat{\boldsymbol{v}}_t^{(b)}, t = 1, \ldots,$ are at least as large as the corresponding elements of $\boldsymbol{u}$. Taking this into account, Algorithm 2.1 always identifies all elements i corresponding to $\delta_i < 0$ as nonbottleneck ones.
2. 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.
3. 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 $\hat{\boldsymbol{v}}$-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.

2.1.1. *Numerical Results*

We now present numerical studies with Algorithm 2.1 for the model (1), (7) consisting of $n \times m$ bridges.

In our numerical results, we assume that the components X_{ijk} of the random vector $\boldsymbol{X}$ are independent each distributed $Weib(\alpha, u)$, 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 Ref.[4] that a Weibull random variable can be generated using the transformation $\beta Z^{1/\alpha}$, where $Z \sim \exp(1)$. We also assume that only u is controllable, while α is fixed and equals 0.2. We purposely selected some elements of $\boldsymbol{u}$ 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 (7), with $\ell = \mathbb{E}_{\boldsymbol{u}} S(\boldsymbol{Y})$, where the sample function $S(\boldsymbol{Y}) = Y_{11}$ and

$$Y_{ij} = \min\begin{Bmatrix} \{X_{111} + X_{114}\}, & \{X_{112} + X_{115}\}, \\ \{X_{111} + X_{113} + X_{115}\}, & \{X_{112} + X_{113} + X_{114}\} \end{Bmatrix}.$$

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

1. "Mean, max, and min $\hat{\ell}$" denote the sample mean, maximum, minimum, and minimal values of the 10 estimates of $\hat{\ell}$.
2. "RE" denotes the sample relative error for $\hat{\ell}$, 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 $\boldsymbol{v}$ 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
	Mean	4.052	3.970	3.734	3.894	3.829
$\hat{\ell}$	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. Also note that for this model, both methods CE and VM identified correctly the two bottleneck parameters.

Table 2 presents a typical dynamics of identifying the two bottleneck parameters at the first stage of Algorithm 2.1 for the above single bridge model having a total of five 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 nonbottleneck and bottleneck one, respectively. One can see that after two replications we have four bottleneck parameters left, after six replications there are three bottleneck parameters left, and after seven replication the process stabilizes identifying correctly the true two bottleneck parameters.

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

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

Table 4 presents performance of Algorithm 2.1 for the 3×10 model with six bottlenecks corresponding to the elements $u_{111}, u_{112}, u_{211}, u_{212}, u_{311}$, and u_{312}. We set $u_{111} = u_{112} = u_{211} = u_{212} = u_{311} = u_{312} = 1$, while the remaining (nonbottlenecks) values are set equal 2. Note again that in this case both CE and VM found exactly the true six bottlenecks.

It follows from the results 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.

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 Typical evolution of the sequence $\{\hat{\boldsymbol{v}}_t\}$ for the VM and VM-SCR methods

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

2.2. The Two-Stage Screening Algorithm for Rare Events

Recall that at the first stage of our two-stage screening algorithm below, we identify the set of the bottleneck parameter vector $\boldsymbol{v}^{(b)}$, while at the second one we estimate the actual values of the components of $\boldsymbol{v}^{(b)}$ using the convex programs (4) and (5).

The main idea of the first stage of our screening algorithm is to identify the bottleneck parameter vector $\boldsymbol{v}^{(b)}$ without involving the LR term W. One may wonder how this can be possible if estimation of the rare-event probability $\ell(\boldsymbol{u}) = \mathbb{E}_{\boldsymbol{u}} I_{\{S(\boldsymbol{X}) \geq \gamma\}}$ by itself is essentially based on IS and, thus on LRs. The trick is to execute the first stage (the screening part) by replacing γ with some $\hat{\gamma}_0$, where $\hat{\gamma}_0$ is the estimator of $(1 - \rho)$ quantile of $S(\boldsymbol{X})$ obtained by simulation under the original pdf $f(\boldsymbol{x}, \boldsymbol{u})$, when $\rho \geq 0.01$. For more details see Ref.[4].

As soon as $\hat{\gamma}_0$ is found from simulation we proceed as follows:

1. Solve the programs (4) and (5) with the advantage that there is no need for the LR terms $W(\boldsymbol{X}, \boldsymbol{u}, \boldsymbol{w})$. For concreteness, we rewrite only (4) by

TABLE 4 Performance of Algorithm 2.1 for the 3×10 model with six bottleneck elements and sample size $N = N_1 = 1000$

Method		CMC	CE	VM	CE-SCR	VM-SCR
	Mean	16.16	16.11	14.84	16.12	15.67
$\hat{\ell}$	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

omitting W's and by replacing γ with $\hat{\gamma}_0$. We have

$$\max_{\boldsymbol{v}} \widehat{D}(\boldsymbol{v}) = \max_{\boldsymbol{v}} \frac{1}{N} \sum_{i=1}^{N} I_{\{S(\boldsymbol{X}_i) \geq \hat{\gamma}_0\}} \ln f(\boldsymbol{X}_i; \boldsymbol{v}), \tag{12}$$

where the sample $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N$ in (12) is taken from the original pdf $f(\boldsymbol{x}, \boldsymbol{u})$. Denote the solution by of (12) by $\hat{\boldsymbol{v}}$. If not stated otherwise, we refer to CE program (12) below.

2. Identify the bottleneck parameter vector $\boldsymbol{v}^{(b)}$ by applying a simple classification procedure, which divides the optimal n-dimensional vector $\hat{\boldsymbol{v}}$ into two parts, namely, as $\hat{\boldsymbol{v}} = (\hat{\boldsymbol{v}}^{(b)}, \hat{\boldsymbol{v}}^{(n)})$. The classification is performed based on the relative perturbation (10), which is the core of the screening algorithm below.

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

1. The same as in Algorithm 2.1.
2. Generate a sample $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N$ from $f(\boldsymbol{x}; \boldsymbol{u})$ and compute $\hat{\gamma}_0$, the $(1-\rho)$-sample quantile of the sample performances $S(\boldsymbol{X}_i)$.
3. Generate a different sample $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N$ from $f(\boldsymbol{x}; \boldsymbol{u})$ and deliver the CE solution of the stochastic program (12), with $\gamma = \gamma_0$. Denote the solution by $\hat{\boldsymbol{v}}_t = (\hat{v}_{t1}, \ldots, \hat{v}_{tn})$. Note that $\hat{\boldsymbol{v}}_t$ is a parameter vector from $f(\boldsymbol{x}; \hat{\boldsymbol{v}}_t)$.

4–8. The same as Steps 3–7 in Algorithm 2.1.

9. Deliver (6) as the resulting estimator of the rare-event probability ℓ.

2.2.1. *Numerical Results*

We next present numerical studies with Algorithm 2.2 for the model (7) 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 nonbottleneck ones.

Table 5 presents the performance of Algorithm 2.2 for the 2×2 model with eight bottlenecks using $\delta = 0.1$, $\gamma = 6$ and the sample sizes $N = 50{,}000$ and $N_1 = 500{,}000$. In particular, we set the six 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 eight bottleneck parameters.

It follows from the results of Table 5 that for this relatively small model both CE and VM perform similarly to their screening counterparts.

TABLE 5 Performance of Algorithm 2.2 for the 2×2 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

We will further see that as the complexity of the model increases VM-SCR outperforms its three 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 2×2 model with 20 parameters, eight of which are bottlenecks. Similar to Table 2, the 0's and 1's in Table 6 mean that an appropriate parameter u is identified as nonbottleneck 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 eight true bottleneck parameters.

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

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

t	$\hat{\gamma}_t$	$\bar{v}_{111}$	$\bar{v}_{112}$	$\bar{v}_{113}$	$\bar{v}_{114}$	$\bar{v}_{115}$
			VM			
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			
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 results of Table 7 that the bottleneck elements decrease more than three times, while the nonbottleneck ones fluctuate about their nominal values equal 4.

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

Table 8 presents the performance of Algorithm 2.2 for the 3×5 model, where we set $u_{111} = u_{211} = u_{311} = 1.5$, $u_{112} = u_{212} = u_{312} = 1$, while

TABLE 8 Performance of Algorithm 2.2 for the 3×5 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
	Mean	1.062E-07	8.728E-08	8.011E-08	8.061E-08
$\hat{\ell}$	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

the remaining (nonbottleneck) 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 2×2 model.

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 the performance of Algorithm 2.2 for the 3×10 model with the above six 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 six bottlenecks.

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

Table 10 presents the performance of Algorithm 2.2 for the model in Figure 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 9 Performance of Algorithm 2.2 for the 3×10 model with six bottlenecks. We set $\delta = 0.1$, $\gamma = 6$, $N = 400{,}000$, $N_1 = 400{,}000$

Method		CE	VM	CE-SCR	VM-SCR
$\hat{\ell}$	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

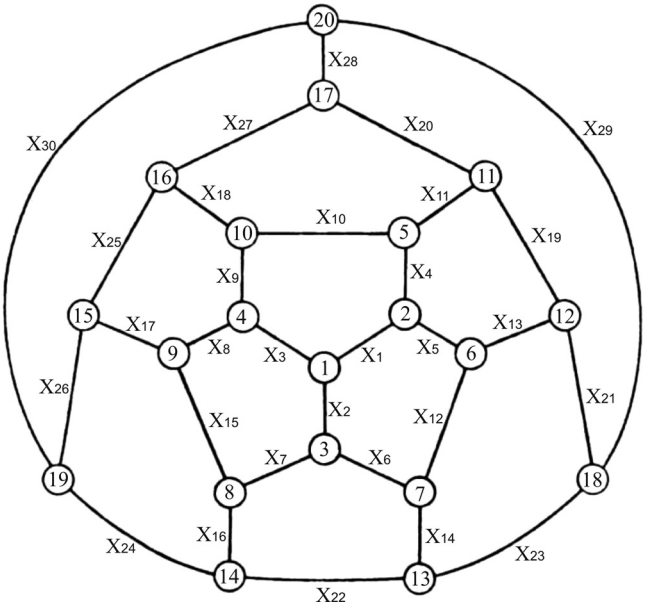

FIGURE 2 A network.

It follows from above that all four approaches perform similarly. The reason why 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 no evident parameters.

While updating the parameter vector $\hat{\boldsymbol{v}}^{(b)}$ at the second stage of Algorithm 2.2 with the above model, we found that six elements of $\hat{\boldsymbol{v}}^{(b)}$ corresponding to the parameters $u_1, u_2, u_3, u_{28}, u_{29}, u_{30}$ have changed 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 six initial bottleneck

TABLE 10 Performance of Algorithm 2.2 for the model in Figure 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
	Mean	1.233E-08	1.234E-08	1.134E-08	1.341E-08
$\hat{\ell}$	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

TABLE 11 Performance of Algorithm 2.2 for the model in Figure 2 with $\delta = 0.1$, $\gamma = 2000$, $N = 100{,}000$, $N_1 = 300{,}000$

Method		CE	VM	CE-SCR	VM-SCR
	Mean	7.415E-09	4.013E-09	4.045E-09	4.331E-09
$\hat{\ell}$	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

parameters $u_1, u_2, u_3, u_{28}, u_{29}, u_{30}$ equal to 1, while increasing the remaining (nonbottleneck) 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 six bottleneck elements.

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

3. SCREENING FOR BERNOULLI MODELS

Note that for Bernoulli random variables, and in particular, while estimating systems unreliability with highly reliable components, that is estimating the following rare-event probability

$$\ell = 1 - \mathbb{E}S(\boldsymbol{X}) = \mathbb{E}I_{\{S(\boldsymbol{X})=0\}}, \tag{13}$$

where $S(\boldsymbol{X})$ is called the structure function, 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 $\boldsymbol{u}$ 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 $f(\boldsymbol{x}, \boldsymbol{u}) = \text{Bern}(\boldsymbol{u})$. Consider, for example, the toy example of estimating $\ell = 1 - \mathbb{E}_u(X)$, where $X \sim \text{Bern}(u)$ and $u = 0.999$. By taking a sample of size $N = 1{,}000$ from such Bern(0.999), we would get on average only one 0, while the rest would be 1's.

To overcome this difficulty, we shall introduce an auxiliary Bernoulli parameter vector $\boldsymbol{p}_0$ (for Bernoulli pdf we shall use the notation $\boldsymbol{p}$ instead of $\boldsymbol{v}$, as in the previous section), which will play the role similar to γ_0 in $\ell(\gamma_0) = \mathbb{E}_{\boldsymbol{u}} I_{\{S(\boldsymbol{X}) \geq \gamma_0\}}$. This can be done, for example, as follows.

Set, say to 0.85 the values of $\boldsymbol{p}_0$ corresponding to $\boldsymbol{u}$, which are greater than 0.85; set the remaining elements of $\boldsymbol{p}_0$ equal to the corresponding elements of $\boldsymbol{u}$. Thus, the elements of the original vector $\boldsymbol{u}$, which are greater than 0.85 are set in $\boldsymbol{p}_0$ automatically to 0.85, while the remaining ones are adopted by $\boldsymbol{p}_0$ without any change. It is readily seen that by doing so $\ell(\boldsymbol{p}_0) = \mathbb{E}_{\boldsymbol{u}} I_{\{S(\boldsymbol{X}) \geq \boldsymbol{p}_0\}}$ will not be a rare-event probability anymore, and thus, while estimating $\ell(\boldsymbol{p}_0)$ we shall generate enough 0's and 1's.

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

1. Select $\boldsymbol{p}_0$ as follows: set to 0.85 the values of $\boldsymbol{p}_0$ corresponding to $\boldsymbol{u}$, which are greater than 0.85; set the remaining elements of $\boldsymbol{p}_0$ equal to the corresponding elements of $\boldsymbol{u}$.
2. The same as Step 1 in Algorithm 2.1.
3. Generate a sample $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N$ from the Bernoulli pdf $f(\boldsymbol{x}, \hat{\boldsymbol{p}}_{t-1}) = \text{Ber}(\hat{\boldsymbol{p}}_{t-1})$ and deliver the solution of the stochastic program (see (12))

$$\max_{\boldsymbol{p}} \widehat{D}(\boldsymbol{p}) = \max_{\boldsymbol{p}} \frac{1}{N} \sum_{i=1}^{N} I_{\{S(\boldsymbol{X}_i)=0\}} W^{(b)}\big(\boldsymbol{X}_i^{(b)}; \boldsymbol{u}^{(b)}, \hat{\boldsymbol{p}}_{t-1}^{(b)}\big) \ln f\big(\boldsymbol{X}_i^{(b)}; \boldsymbol{p}\big). \tag{14}$$

   Denote the solution by $\hat{\boldsymbol{p}}_t = (\hat{p}_{t1}, \ldots, \hat{p}_{tk})$. Note also that in contrast to (12), $W(\boldsymbol{X}_i, \boldsymbol{u}, \boldsymbol{p}_0) = 1$ *only* for the elements of $\boldsymbol{p}_0$ which were set *equal* to $\boldsymbol{u}$.
4. Calculate the relative perturbation for each element $\hat{p}_{tr}, r = 1, \ldots, k$ as

$$\delta_{tr} = \frac{u_r - \hat{p}_{tr}}{u_r}. \tag{15}$$

5–8. The same as Steps 4–7 in Algorithm 2.1.

9. Estimate ℓ in (13) (see also (6)) as

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

3.1. Numerical Results

We start with the numerical results corresponding to the reliability system for the model in Figure 1 when the sample function $S(\boldsymbol{X})$ is defined in (8). Similar to the tables in section 2.2, we consider here models with more evident and less evident bottlenecks. Clearly, 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 2.2:

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 nonbottleneck ones.

Table 12 presents the performance of Algorithm 3.1, for the 2×2 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 a model as the one with evident bottleneck elements. Clearly, in this case there are eight bottleneck elements, corresponding to $u_{ijk} = 0.99$. All eight bottleneck elements were identified correctly by both CE and VM methods.

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

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

We next present simulation results for larger models, namely, for 3×5 and 3×10 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 six bottlenecks, which were correct by the CE and VM methods.

Table 14 presents the performance of the Algorithm 3.1 for the 3×5 model with $N = 50{,}000$ and $N_1 = 250{,}000$ samples.

Table 15 presents similar data for 3×10 model. We set $N = 100{,}000$, $N_1 = 300{,}000$.

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.

TABLE 12 Performance of Algorithm 3.1 for the 2×2 model with evident bottlenecks. 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 Typical evolution of sequence $\{\hat{\boldsymbol{p}}_t\}$

			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

TABLE 14 Performance of Algorithm 3.1 for the 3×5 model with $N = 50{,}000$ and $N_1 = 250{,}000$ samples

Method		CE	VM	CE-SCR	VM-SCR
	Mean	2.983E-08	3.450E-08	2.257E-08	2.146E-08
$\hat{\ell}$	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 Performance of Algorithm 3.1 for the 3×10 model with samples $N = 100{,}000$, $N_1 = 300{,}000$

Method		CE	VM	CE-SCR	VM-SCR
	Mean	4.583E-08	5.909E-08	4.096E-08	4.699E-08
$\hat{\ell}$	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

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
	Mean	0.000E+0	2.232E-6	2.073E-6	1.000E-6	2.048E-6
$\hat{\ell}$	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

We finally consider again the model from Ref.[2] depicted in Figure 2.

Table 16 presents the 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 $\hat{\boldsymbol{p}}^{(b)}$ at the second stage of Algorithm 3.1, we found that the six elements of $\hat{\boldsymbol{p}}^{(b)}$ corresponding to the parameters $u_1, u_2, u_3, u_{28}, u_{29}, u_{30}$ changed the most and were correctly identified by the VM method. Note that CE identified only three bottlenecks. Because of this CE-SCR performs poorly as Table 16 shows.

Taking this into consideration that six parameters $u_1, u_2, u_3, u_{28}, u_{29}, u_{30}$ changed most we can easily modify the initial parameter vector $\boldsymbol{u}$ 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 six initial bottleneck parameters $u_1, u_2, u_3, u_{28}, u_{29}, u_{30}$, while keeping the remaining (nonbottleneck) 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 six 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
	Mean	6.000E-5	4.353E-5	5.436E-5	5.436E-5	5.442E-5
$\hat{\ell}$	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[5]. 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 we not only 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 three 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 LRs. 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.

ACKNOWLEDGMENT

This research was supported by the Binational Science Foundation.

REFERENCES

1. Asmussen, S.; Glynn, P. *Stochastic Simulation*; Springer: New York, 2007.
2. Fishman, G. *Monte Carlo*; Springer: New York, 1995.
3. Liu, J.S. *Monte Carlo Strategies in Scientific Computing*; Springer: New York, 2001.
4. Rubinstein, R.Y.; Kroese, D.P. *The Cross-Entropy Method: A Unified Approach to Combinatorial Optimization, Monte-Carlo Simulation and Machine Learning*; Springer: New York, 2004.
5. Rubinstein, R.Y.; Kroese, D.P. *Simulation and the Monte Carlo Method*, 2nd Ed.; Wiley: New York, 2007.