Splitting with Weight Windows to Control the Likelihood Ratio in Importance Sampling

(Invited paper)

Pierre L'Ecuyer
DIRO, Université de Montréal
C.P. 6128, Succ. Centre-Ville
Montréal, H3C 3J7, Canada
lecuyer@iro.umontreal.ca

Bruno Tuffin
IRISA-INRIA, Campus Universitaire de Beaulieu
35042 Rennes Cedex, France
Bruno.Tuffin@irisa.fr

ABSTRACT

Importance sampling (IS) is the most widely used efficiency improvement method for rare-event simulation. When estimating the probability of a rare event, the IS estimator is the product of an indicator function (that the rare event has occurred) by a likelihood ratio. Reducing the variance of that likelihood ratio can increase the efficiency of the IS estimator if (a) this does not reduce significantly the probability of the rare event under IS, and (b) this does not require much more work. In this paper, we explain how this can be achieved via weight windows and illustrate the idea by numerical examples. The savings can be large in some situations. We also show how the technique can backlash when the weight windows are wrongly selected.

Categories and Subject Descriptors

I.6 [**Computing Methodologies**]: Simulation and Modeling

General Terms

Algorithms, Reliability

1. INTRODUCTION

Estimating the probability of a rare event has several applications in reliability, telecommunications, insurance, and several other areas. For complicated models, this can be done in principle by Monte Carlo simulation, but when the event of interest is really rare, straightforward simulation would usually require an excessive number of runs for the rare event to happen frequently enough so that the estimator is meaningful.

Importance sampling (IS) is a technique that changes the probability laws that drive the model's evolution, to increase the probability of the rare event. The estimator is then multiplied by an appropriate likelihood ratio so that it has the correct expectation. Details can be found in [3,14,15,17], and other references given there.

A second technique that deals with rare events is *splitting and Russian roulette* (often simply called *splitting*) [2, 7–10, 13, 18, 19, 25]. It does not change the probability laws that drive the model, but it creates an attraction toward the rare event(s) by a applying a *selection* mechanism to the sample paths: A trajectory (or sample path) that seems to go away from the rare event is terminated with some probability (the weight of the any survivor is multiplied by this probability; this is *Russian roulette*) and those that seem to go in the right direction are *split* (or cloned, and the weight is divided equally among them).

In this paper, we are interested in combining the two methods as follows: We want to use splitting to control the variance of the likelihood ratio when applying IS. In our setting, the estimator is an indicator function (it indicates if the rare event has happened or not) multiplied by a likelihood ratio. When the rare event happens, the likelihood ratio is usually much smaller than 1 and it sometimes has a large variance. In that case, reducing this variance could bring significant efficiency improvement in addition to that obtained from IS. Booth [1, 2] suggested the idea of *splitting with weight windows* to reduce this variance. The *weight* of a sample path at any given time is redefined as the product of its weight from the splitting and Russian roulette by the likelihood ratio that it has accumulated so far. When a sample path reaches the rare event, this weight is its contribution to the estimator. Ideally, two sample paths that have about the same probability of reaching the rare event (conditional on their state) should have about the same weight. The idea is to figure out what this weight should be, on average, as a function of the state, and define a window (an interval) around this average, which can be viewed as a target for the weight. Whenever the weight of a sample path falls below the window, we apply Russian roulette; the path is killed with some probability $1 - p_s$ and its weight is multiplied by $1/p_s$ if it survives. The value of p is selected to that the new weight falls in the window. When the weight goes above the window, we split the sample path in a number of copies so that the weight of each copy is inside the window. In the proposal of [1, 2], the state space is partitioned in a finite number of regions and each region has a different weight window (constant over the region). Such weight windows

Valuetools '06, October 11-13, 2006, Pisa, Italy.

are used in the Los Alamos particle transport simulation programs. The weight windows are selected by the user, or by heuristics, or by using prior runs to estimate what should be the average weight of a sample path in any given region, if this path reaches the rare event.

The remainder of the paper is organized as follows. Our basic rare-event setting is defined in Section 2. In Section 3, we describe how to combine IS with weight windows based on the *weighted importance*, defined as the product of the weight by a function of the state called *importance function*. With the right choice of importance function, the expected weighted importance should be a constant, independent of the state, and we center our weight window (which is on the weighted importance and does not depend on the state) around this constant. In Section 4, we give numerical illustrations. In one of the examples, we show what can go wrong with a bad guess of the importance function.

2. SETTING

We consider a discrete-time Markov chain $\{X_j,\, j \geq 0\}$ with state space $\mathcal{X}$. Let A and B be two disjoint subsets of $\mathcal{X}$. The chain starts in some state $X_0 = x$, leaves the set A if $x \in A$, and then eventually reaches B or A. Let

$$\tau_A = \inf\{j > 0 : X_{j-1} \in A \text{ and } X_j \in A\}$$

and

$$\tau_B = \inf\{j \geq 0 : X_j \in B\}.$$

For all $x \in \mathcal{X}$, let

$$\gamma(x) = \mathbb{P}[\tau_B < \tau_A \mid X_0 = x], \tag{1}$$

which is the probability that the chain reaches the rare event before getting back to A if it starts in state x. We have $\gamma(x) = 1$ if $x \in B$. We also denote $\gamma = \gamma(x_0) = \mathbb{P}[\tau_B < \tau_A]$, where x_0 is some fixed initial state. We want to estimate γ. This form of rare-event problem, where γ is small, occurs in many practical situations; see, e.g., [4, 17, 21, 22].

The standard (or crude) Monte Carlo method to estimate γ is to simulate the chain n times until it reaches the stopping time $\tau = \min[\tau_A, \tau_B]$, independently, and define the estimator $\hat{\gamma}_n$ as the proportion of those n runs for which the event $\{\tau_B < \tau_A\}$ occurs. This estimator has *relative error*

$$\mathrm{RE}[\hat{\gamma}_n] = \frac{(\mathrm{Var}[\hat{\gamma}_n])^{1/2}}{\gamma} = \frac{(\gamma(1-\gamma)/n)^{1/2}}{\gamma} = O\left(\frac{1}{(\gamma n)^{1/2}}\right)$$

when $\gamma \to 0$. For a bounded relative error, this estimator requires a n that increases as $O(1/\gamma)$, so it eventually becomes too time-consuming to be practical.

To introduce IS for this model, we consider the special case of a Markov chain with denumerable state space $\mathcal{X}$. The same ideas apply if the state space is continuous; the transition probabilities can then be replaced by conditional densities. Let $p(x, y) = \mathbb{P}[X_j = y \mid X_{j-1} = x]$ be the transition probabilities, for $x, y \in \mathcal{X}$. With IS, these probabilities are changed to $q(x, y)$, for $x, y \in \mathcal{X}$, where $q(x, y) > 0$ whenever $p(x, y)\gamma(y) > 0$. To compute the IS estimator, we simulate the chain with the transition probabilities $q(x, y)$ instead of $p(x, y)$, and the (unbiased) estimator of γ (for each run) becomes:

$$\hat{\gamma}_{\mathrm{is}} = \mathbb{I}[\tau_B < \tau_A] L(X_1, \ldots, X_\tau), \tag{2}$$

where

$$L(X_1, \ldots, X_\tau) = \prod_{j=1}^{\tau} \frac{p(X_{j-1}, X_j)}{q(X_{j-1}, X_j)} \tag{3}$$

is the likelihood ratio associated with the probability change. For n independent simulation runs, we just average the n copies of $\hat{\gamma}_{\mathrm{is}}$; the corresponding estimator is denoted $\hat{\gamma}_{\mathrm{is},n}$.

There are two sources of variance in the estimator (2): the indicator function and the likelihood ratio. Without IS, the likelihood ratio is 1 and all the variance is in the indicator function, which is nonzero only with a very small probability. With IS, the indicator function should be 1 with much larger probability (if the probabilities $q(x, y)$ are well-chosen), but the likelihood ratio can also vary. If this likelihood ratio was a constant when the rare event occurs, then at least that contribution to the variance would be eliminated and there would only remain the variance of the (binomial) indicator random variable, multiplied by some constant. One of our goals here is to approximate this by making the (weighted) likelihood ratio almost a constant when B is reached. We also want to see if we can do better by using splitting to reduce the variance of both the likelihood ratio and the indicator simultaneously.

3. WEIGHT WINDOWS FOR IMPORTANCE SAMPLING

3.1 Controlling the likelihood ratio.

Let $\mathbb{Q}$ denote the probability measure under IS, let $\mathbb{E}_{\mathrm{q}}$ and $\mathrm{Var}_{\mathrm{q}}$ denote the corresponding expectation and variance operators, and let

$$\gamma_{\mathrm{q}}(x) = \mathbb{Q}[\tau_B < \tau_A \mid X_0 = x], \tag{4}$$

the probability that the chain reaches B before getting back to A if it starts in state x, under IS. Note that

$$\begin{aligned}
\gamma(x) &= \mathbb{E}_{\mathrm{q}}[\hat{\gamma}_{\mathrm{is}} \mid X_0 = x] \\
&= \mathbb{E}_{\mathrm{q}}[L(X_1, \ldots, X_\tau) \mid \tau_B < \tau_A,\, X_0 = x] \\
&\quad \times \mathbb{Q}[\tau_B < \tau_A \mid X_0 = x] \\
&= \ell(x)\gamma_{\mathrm{q}}(x)
\end{aligned}$$

where

$$\ell(x) = \mathbb{E}_{\mathrm{q}}[L(X_1, \ldots, X_\tau) \mid \tau_B < \tau_A,\, X_0 = x].$$

When starting in state x, the IS estimator has variance

$$\begin{aligned}
&\mathrm{Var}_{\mathrm{q}}[\hat{\gamma}_{\mathrm{is}} \mid X_0 = x] \\
&= \mathbb{E}_{\mathrm{q}}[\hat{\gamma}_{\mathrm{is}}^2 \mid X_0 = x] - \gamma^2(x) \\
&= \mathbb{E}_{\mathrm{q}}[L^2(X_1, \ldots, X_\tau) \mid \tau_B < \tau_A,\, X_0 = x] \\
&\quad \times \mathbb{Q}[\tau_B < \tau_A \mid X_0 = x] - \gamma^2(x) \\
&= \mathbb{E}_{\mathrm{q}}[L^2(X_1, \ldots, X_\tau) \mid \tau_B < \tau_A,\, X_0 = x] \\
&\quad \times \gamma_{\mathrm{q}}(x) - \gamma^2(x) \\
&\geq \ell^2(x)\gamma_{\mathrm{q}}(x) - \gamma^2(x) \qquad (5) \\
&= \gamma^2(x)(1 - \gamma_{\mathrm{q}}(x))/\gamma_{\mathrm{q}}(x).
\end{aligned}$$

In (5), equality holds if and only if

$$\mathbb{Q}[L(X_1, \ldots, X_\tau) = \ell(x) \mid \tau_B < \tau_A,\, X_0 = x] = 1, \tag{6}$$

i.e., if the likelihood ratio is a constant (that depends only on x) when the rare event occurs. Making the likelihood ratio a constant does not necessarily minimize the variance

(because we could also increase γ_q), but it could reduce it substantially. Finding transition probabilities q so that (6) holds for the Markov chain under IS is rarely possible. But we can use splitting to make the likelihood ratio (or a weighted version of it) almost a constant, as follows.

Suppose for a moment that (6) holds for all x, that the chain starts in state $X_0 = x_0$, and that after j steps, for $j \le \tau$, it is in state $X_j = x$. It follows from (6) that if $\tau_B < \tau_A$, then with probability 1, $L(X_1, \ldots, X_\tau) = \gamma(x_0)/\gamma_q(x_0)$ and $L(X_j, \ldots, X_\tau) = \gamma(x)/\gamma_q(x)$, which implies that

$$L(X_1, \ldots, X_j) \;=\; \frac{L(X_1, \ldots, X_\tau)}{L(X_j, \ldots, X_\tau)} = \frac{\ell(x_0)}{\ell(x)}. \qquad (7)$$

This means that if we are in state $X_j = x$, the likelihood ratio accumulated so far should obey the equality (7), ideally. What can we do if it does not?

In fact, $L(X_1, \ldots, X_j)$ can be viewed as a *weight* that the chain has accumulated so far. This weight will simply multiply the contribution of this chain to the estimator at the end (if $\tau_B < \tau_A$) and otherwise has no influence on the sample path of the chain after step j. If we decide to apply splitting or roulette to this chain at step j, then the weighting factors that these methods introduce can simply multiply the likelihood ratio. We will denote the product of these two types of weights by W_j at step j. Thus, we have $W_0 = 1$ and

$$W_j = W_{j-1} R_j p(X_{j-1}, X_j)/q(X_{j-1}, X_j)$$

for $j \ge 1$, where $R_j = 1/p_s$ if we kill the chain with probability $1 - p_s$ at step j (we apply Russian roulette) and the chain survives, $R_j = 0$ if the chain does not survive, and $R_j = 1/\mathbb{E}[C]$ if we split the chain in C copies at step j (where C can be a random variable). In the latter case, the weight W_j is given to each copy of the chain. We take C as a random variable when we have a non-integer target value c for $\mathbb{E}[C]$; we then take $C = \lfloor c \rfloor + 1$ with probability $\delta = c - \lfloor c \rfloor$ and $C = \lfloor c \rfloor$ with probability $1 - \delta$. If we start with n chains in state x_0, then the (unbiased) estimator $\hat{\gamma}_{\mathrm{is},n}$ of $\gamma(x_0)$ is now the sum of all the weights when the chains have reached their stopping times τ, divided by n.

To mimic (7), we would like that

$$Y_j = W_j \ell(X_j)/\ell(x_0) \approx 1 \qquad (8)$$

at all steps. To make this happen, we suggest to select a window $[\alpha_1, \alpha_2]$ for some real numbers $\alpha_1 < 1 < \alpha_2$. For example, we could take $\alpha_1 = 1/2$ and $\alpha_2 = 2$. Whenever $Y_j > \alpha_2$, we split the chain in C copies where $\mathbb{E}[C] = Y_j$, and whenever $Y_j < \alpha_1$, we apply Russian roulette, killing the chain with probability $1 - p_s = 1 - Y_j$. Note that it is possible that doing this may decrease the fraction of chains that reach B, in which case the variance is not necessarily reduced.

Our development so far assumes that the goal of splitting is only to keep the likelihood ratio close to its conditional expectation, and not necessarily to favor the chains that are getting closer to B. We rely only on IS to do the "pushing" toward B. In fact, without IS, $\ell(x) = 1$ for all x, so with the previous method all the chains would keep weight 1 and no splitting or roulette would occur.

3.2 Controlling the weighted importance.

When splitting is used alone, it is known that the best thing to do (in terms of variance) is to make sure that all the chains have approximately the same expected contribution to the estimator [19]. If a chain is in state $X_j = x$ and has weight W_j, its expected contribution is $W_j \gamma(x)$, with or without IS. Initially, the chain is in state x_0 and has weight 1, so its expected contribution is $\gamma(x_0)$. Thus, a second viewpoint would be that we want

$$Y_j = W_j \gamma(X_j)/\gamma(x_0) \approx 1 \qquad (9)$$

at all steps, instead of (8). The splitting and roulette then helps pushing the chain toward B in addition to controlling the likelihood ratio. We can apply the weight windows in the same way, with (9) instead of (8), and the estimator has the same definition. With either method, the chains evolve independently of each other and the number of chains at any given point in time is random.

3.3 Fixed number of chains.

Suppose now that we want the number of chains to be always equal to some fixed integer n, minus the number n_B of chains that have reached the set B so far. Thus, the number of chains that reach B will always be equal to n or slightly smaller. It can be smaller because it may happen that all the $n - n_B$ chains alive at a given step hit A in the next transition. So at any given step, we want to have $n - n_B$ chains with approximately the same weighted importance. If chain i is in state $X_{i,j}$ and has weight $W_{i,j}$ at step j, counting only the chains that have not reached their stopping time, we should replace (9) by

$$Y_{i,j} \;=\; W_{i,j}\gamma(X_{i,j})/\gamma(x_0) \qquad (10)$$

$$\approx\; \frac{1}{n - n_B} \sum_{i=1}^{n-n_B} W_{i,j}\gamma(X_{i,j})/\gamma(x_0) \qquad (11)$$

for all i. That is, the expected contribution should be divided equally among the $n - n_B$ chains. A similar adaptation can be defined for (8). To implement the method, we select a fixed constant $\alpha > 1$. At each step, we simulate one transition for all the chains, then we split in two copies the chain with the largest value of $Y_{i,j}$ in (10), repeatedly, until we have $n - n_B$ chains. After that, as long as α times the smallest value of $Y_{i,j}$ is less than the largest value of $Y_{i,j}$, we apply roulette to the chain with the smallest value so that its weight is raised up to the average in (11) if it survives. If it does not survive, then we split in two the chain with the largest value. The estimator of $\gamma(x_0)$ is the same as before.

3.4 Multilevel setting.

So far we have considered splitting without levels. In the usual *multilevel splitting* procedure [9, 12], we select m levels $\gamma(x_0) < \ell_1 < \cdots < \ell_m = 1$ and simulate n chains independently from state x_0 until all these chains have reached either A or a state X_j with $\gamma(X_j) \ge \ell_1$. Let R_1 be the number of chains for which the latter happens; these chains are said to have reached level ℓ_1. In the *fixed-effort* and *fixed-assignment* version of multilevel splitting, these R_1 chains are split into n chains as follows: If $c = \lfloor n/R_1 \rfloor$ and $d = n \bmod R_1$, we select d of the R_1 chains at random, without replacement; the selected chains are split in $c + 1$ copies and the other ones are split in c copies. This is the

first stage. In stage k, we start with n chains obtained after splitting (using the procedure just described) the R_{k-1} chains that have reached level ℓ_{k-1} and we simulate them independently (from then on) until they reach either A or level ℓ_k. Let R_k be the number of chains that reach ℓ_k. Then it can be shown [9] that $\hat{\gamma}_{\mathrm{is},n} = R_1 R_2 \cdots R_m / n^m$ is an unbiased estimator of $\gamma(x_0)$.

To combine this method with IS and weight windows, we can do the splitting and roulette based on (10), but do it only at the end of each stage of the multilevel splitting procedure. Suppose R_k chains have reached level ℓ_k in stage k. We want to split/kill those chains so that we obtain n chains having approximately the same value of $Y_i = W_{i,j}\gamma(X_{i,j})/\gamma(x_0)$ at the beginning of stage $k+1$. One simple way to achieve this (approximately) is as follows. We split the chain with the largest value of $Y_i = W_{i,j}\gamma(X_{i,j})/\gamma(x_0)$ and repeat this until there are n chains. Then, as long as the largest Y_i is more than twice the smallest Y_i, we apply roulette to the chain with the smallest Y_i, killing it with probability $1 - Y_i/\bar{Y}_n$ where $\bar{Y}_n$ is the average of the Y_i's, and if this chain is killed then we split the chain with the largest Y_i. When this procedure stops, we have exactly n chains with similar values of Y_i, and we can start the next stage.

3.5 Approximating γ by an Importance Function.

We have described all the splitting methods in an *idealistic setting* where the function γ would be known everywhere. If this was true, there would be no need to do simulation to estimate $\gamma(x_0)$ in the first place. In practice, for the splitting and roulette, the function γ is replaced by some approximating function h usually called the *importance function*. A good choice of h is very important for the splitting to be effective [11], but a rough guess of the true function γ is often sufficient, especially in the context of multilevel splitting, provided that we have a good idea of its correct shape. When weight windows are used with splitting without levels, we need a good approximation of at least the order of magnitude of γ in all the area of the state space that we are likely to visit.

4. EXAMPLES

We consider a highly-reliable Markovian system (HRMS) [22] with c types of components and n_i components of type i, for $i = 1, \ldots, c$. Suppose that $\{Y(t) = (Y_1(t), \ldots, Y_c(t)),\ t \geq 0\}$ is a *continuous-time Markov chain*, where $Y_i(t) \in \{0, \ldots, n_i\}$ is the number of *failed* components of type i at time t. Each transition corresponds to the failure of an operational component or the repair of a failed component. Let $\xi_0 = 0$, let $0 \leq \xi_1 \leq \xi_2 \leq \cdots$ be the jump times of this chain, and let $\{X_j,\ j \geq 0\}$ be the embedded discrete-time Markov chain, defined by $X_j = Y(\xi_j)$ for $j \geq 0$. We denote its transition probabilities by $p(x,y) = \mathbb{P}[X_j = y \mid X_{j-1} = x]$.

The set $A = \{x_0\}$ contains only the state $x_0 = (0, \ldots, 0)$ in which all components are operational and B is the set of states in which the system is down. Suppose that the system is up when at least m_i components of type i are operational for each i, for some positive integers m_i; otherwise the system is down. When $\gamma = \mathbb{P}[\tau_B < \tau_A]$ is very small, its estimation is the most challenging part in the estimation of other quantities such as the mean time to failure or the steady-state availability of the system [4,16,22]. To estimate γ, we focus on the simulation of the discrete-time Markov chain $\{X_j,\ j \geq 0\}$.

Using a simplified version of the model of Shahabuddin [22], we assume that each type-i component has *repair rate* μ_i when it is down and *failure rate* $\lambda_i = a_i \varepsilon^{b_i}$ when it is operational, where $a_i > 0$ and b_i is a strictly positive integer, whereas the *rarity parameter* ε satisfies $0 < \varepsilon \ll 1$. The constants a_i, b_i, and μ_i do not depend on ε. Several heuristics have been proposed to select a change of measure for IS in this setting [4, 20]; the most robust approach with respect to rarity is the *balanced failure biasing* (BFB) scheme of [22], which has bounded relative error when $\varepsilon \to 0$. See [5, 20, 23, 24] for further discussions.

BFB replaces the probabilities $p(x,y)$ by new transition probabilities $q(x,y)$ defined as follows. For any state $x \in \mathcal{X}$, let $F(x)$ [resp., $R(x)$] be the set of states y such that the direct transition $x \to y$ (if any) corresponds to a failure [resp., a repair]. Let $p_{\mathrm{F}}(x) = \sum_{y \in F(x)} p(x,y)$ and $p_{\mathrm{R}}(x) = \sum_{y \in R(x)} p(x,y)$ be the overall failure and repair probabilities, respectively. We select a constant $\rho \in (0,1)$, not too close to 0 or 1. For a state $x \notin B$, we define

$$q(x,y) = \begin{cases} \frac{1}{|F(x)|} & \text{if } y \in F(x) \text{ and } p_{\mathrm{R}}(x) = 0; \\ \rho \frac{1}{|F(x)|} & \text{if } y \in F(x) \text{ and } p_{\mathrm{R}}(x) > 0; \\ (1-\rho)\frac{p(x,y)}{p_{\mathrm{R}}(x)} & \text{if } y \in R(x); \\ 0 & \text{otherwise.} \end{cases}$$

For $x \in B$, we take $q(x,y) = p(x,y)$. This change of measure increases the probability of failure to at least ρ when the system is up, so a failure-induced transition is no longer a rare event. The new failure rate is split equally between all states that can be reached directly from the current state by a failure. The BFB estimator of γ and the likelihood ratio are defined as in (2) and (3).

EXAMPLE 1. *For our first numerical illustration, we take a simple system with 10 components of the same type, so $c = 1$ and $n_1 = 10$. The other parameters values are $m_1 = 2$, $\mu_1 = 1$, $\varepsilon = 0.05$, $\lambda_1 = 0.001$, and $\rho = 0.5$. For this example, we have $\gamma \approx 9.0 \times 10^{-24}$ and the variance per run with BFB is $\sigma^2_{\mathrm{is}} \approx 2.0 \times 10^{-44}$. Thus, the relative error with n runs is approximately $15.7/\sqrt{n}$.*

To define an importance function h that approximates γ, we first observe that for any choice of $\rho < 1$, $\gamma(x)$ cannot exceed the largest value of the likelihood ratio that can be obtained over all sample paths going from state x to the set B. This upper bound is smallest when $\rho \to 1$ (in the limit), so we define $h(x)$ as its value when $\rho = 1$ (i.e., we take the tightest upper bound). This gives:

$$\frac{\gamma(x)}{\gamma(x_0)} \approx \frac{h(x)}{h(x_0)} = \prod_{j=1}^{x-1} \left(1 + \frac{j\mu_1}{(n_1 - j)\lambda_1}\right).$$

We applied the window $[\alpha_1, \alpha_2] = [0.5, 2.0]$ to the weighted importance (9), separately (independently) for each chain, using this approximation for γ. We started with n chains and computed the estimator $\hat{\gamma}_{\mathrm{is},n}$ defined as the sum of weights

of all chains that reached B, divided by n. This was replicated m times, independently. The mean and variance per run were estimated by the average of the m realizations of $\hat{\gamma}_{\text{is},n}$ and n times their empirical variance, respectively. We denote these two quantities by $\hat{\gamma}_{\text{is,w}}$ and $\hat{\sigma}^2_{\text{is,w}}$. Their expectations do not depend on n and m. In a simulation experiment with $n = 4$ and $m = 2^{20}$, we obtained $\hat{\gamma}_{\text{is,w}} = 9.0 \times 10^{-24}$ and $\hat{\sigma}^2_{\text{is,w}} = 8.6 \times 10^{-46}$, so using the weight windows reduced the variance (empirically) by a factor of 23. The number of hits of B per initial chain was about 0.11 with BFB alone and 0.51 with the weight windows.

To compare the work required by BFB with and without the weight windows, we can look at the average number of simulated steps of the Markov chain, per original chain, for each of the two methods. The inverse of the product of this work by the variance per run can be taken as a measure of efficiency. The average number of simulated steps was 9.0 with BFB alone and 7.6 with the weight windows. So not only the variance is reduced but the work (measured by the number of simulated steps) is also reduced. Note that here, we do not account for the overhead required by the splitting and roulette, which depends heavily on the implementation.

We then tried the same example with $n_i = 20$ instead of $n_i = 10$. Here, $\gamma \approx 1.9 \times 10^{-53}$. The variance was reduced (empirically) by a factor of 2409 and the average number of simulated steps per initial chain was 19.0 for BFB alone and 15.6 for BFB with the weight windows.

Finally, returning to $n_1 = 10$, we tried the method with a fixed number of chains*, with $n = 1024$ chains, $m = 1024$ independent replications, and $\alpha = 4.0$. The variance was reduced approximately by a factor of 29. The average number of steps per initial chain was 11.1 and the average number of hits of B per initial chain was 0.9998. Here we have more hits and more variance reduction than with the previous method, but also more work, and the efficiency is roughly the same.*

EXAMPLE 2. *We now consider a system with $c = 3$ component types, $n_i = 6$, $m_i = 2$, and $\mu_i = 1$ for all i, $\lambda_1 = \varepsilon$, $\lambda_2 = 1.5\varepsilon$, $\lambda_3 = 2\varepsilon^2$, $\varepsilon = 0.01$, and $\rho = 0.8$. Here, $\gamma \approx 1.8 \times 10^{-7}$ and the variance per run with BFB is $\sigma^2_{\text{is}} \approx 1.0 \times 10^{-11}$. For system's failure, we need at least $n_i - m_i + 1 = 5$ failures of the same type of component.*

We made a simulation experiment as in the previous example, with $n = 4$ and $m = 2^{20}$. The function γ was also approximated by the largest likelihood ratio that can be obtained over all sample paths going from state x to the set B, with $\rho = 1$, and we used the window $[\alpha_1, \alpha_2] = [0.5, 2.0]$ for (9). We obtained $\hat{\gamma}_{\text{is,w}} = 1.8 \times 10^{-7}$. The variance was reduced (empirically) by a factor of 7.1 only. However, the average number of steps per initial chain was also reduced from 11.2 to 3.3. Interestingly, the average number of hits of B per initial chain was reduced from 0.75 with BFB alone to 0.0089 with the weight windows. This suggests that our approximation $h(x)/h(x_0)$ grossly underestimates the ratio $\gamma(x)/\gamma(x_0)$ when x approaches the set B, so the weighted importance becomes much smaller than what it should be, and this results in killing too many of the chains having a large number of failed components.

To verify this intuition, we computed $h(x_0) \approx 73.2 \times 10^{-7}$, a much larger value than $\gamma(x_0) \approx 1.8 \times 10^{-7}$. On the other hand, $h(x) = \gamma(x) = 1$ whenever $x \in B$. We thus have $h(x)/h(x_0) = \gamma(x)/\gamma(x_0) = 1$ for $x = x_0$, but $h(x)/h(x_0) \ll \gamma(x)/\gamma(x_0)$ for $x \in B$. Along the paths from x_0 to B, $h(x)/h(x_0)$ is generally much smaller than it should be when x is close to B and not much smaller when x is near x_0.

Why is that? With the BFB adopted here, the shortest path to failure (with $m_i + 1$ failures for type-i components and no other failure) does not represent the typical path to failure. In a typical path, there is usually a few failures of other components as well, and often a few repairs. This reduces the (average) final value of the likelihood ratio and this is why $h(x_0)$ overestimates $\gamma(x_0)$ (which equals the expected value of the likelihood ratio times the indicator that we reach B).

If this explanation is correct, then the underestimation of $\gamma(x)/\gamma(x_0)$ should get much worst if we increase n_i and m_i while keeping the other parameters unchanged. This is confirmed by the next example.

EXAMPLE 3. *We take the same parameters as in the previous example, except that we now try $n_i = 8$, 10, and 12. As n_i increases, the overestimation of $\gamma(x_0)$ gets worse very quickly:*

n_i	8	10	12
$\gamma(x_0)$	6.0×10^{-11}	1.7×10^{-14}	4.2×10^{-18}
$h(x_0)$	1.5×10^{-8}	3.0×10^{-11}	6.0×10^{-14}

As a result, since $\gamma(x)/\gamma(x_0)$ is grossly underestimated, when x approaches B the chains are killed with high probability at each step, because their weighted importance is almost always much lower than the weight window. Then, very few chains (if any) can reach the target set B and the weight windows are doing more damage than good. In fact, in an experiment with $n_i = 12$ and a fixed *number of chains, no chain ever made it to B even after several hours of CPU time. The chains were constantly killed as they were approaching B and were replaced by chains that were closer to x_0, so the weight windows were in fact* preventing *the chains from reaching B.*

How can we avoid this bad behavior and make sure that the weight windows really help? One way is to first get some rough estimate of the function γ via pilot runs, and use this estimate to define the weight windows.

For the case of $n_i = 12$ above, for example, we could raise the current h to the power $\alpha \approx 18.2/14$ to get a new h that better matches γ. With this new h, the weight windows reduced the variance per run from 1.1×10^{-29} to 2.1×10^{-34}; that is, a spectacular reduction factor of more than 53000. On the other hand, the average number of steps per initial chain increased from 30 to 335 and the average number of hits per initial chain decreased from 0.75 to 0.13. With a fixed number of chains ($n = 1024$) and the same h for the weight windows, we observed a variance reduction factor of approximately 6200, while the average number of steps and the average number of hits per initial chain were 46.4 and 1.0, respectively.

Another (potential) remedy is to adopt a different failure biasing scheme that gives higher probabilities to the sample paths where all failures are for the same type of component. To illustrate this, we consider an adaptation of the distance-based biasing (DBB) scheme of [6], which (adaptively) increases the failure probabilities of component types when more components of that type are already failed. DBB allocates a fixed probability $1-\rho$ to repair transitions (as for BFB), and splits this probability to the different repair types in proportion to their original probabilities. There remains a probability ρ for the failure transitions. A fraction ρ_c of this probability (that is, a probability $\rho_c\rho$) is allocated to a failure transition of component type i type with the smallest value of $(n_i-m_i-x_i+1)b_i$, i.e., which gives the largest probability of failure if all failures are for the same type of component (in case of equality, this probability is split equally among the component types that reach the minimum). The same procedure is applied recursively to the remaining component types. That is, a fraction ρ_c of the remaining probability is allocated to a failure transition of component type i type with the second smallest value of $(n_i-m_i-x_i+1)b_i$, and so on. In each subgroup, the individual probabilities are taken proportional to the number of operational components of the given type. Our modification to the original scheme of [6] is that we consider the probability of paths instead of just the number of transitions required for failure. We also balance the transition probabilities, leading to bounded relative error [4].

For the present example with $n_i = 12$, this method gave a variance per run of 2.2×10^{-33} without the weight windows and 2.9×10^{-35} with the weight windows, with an average number of steps and average number of hits per chain of 16.4 and 0.75 without weight windows, and 4853 and 63.5 with weight windows. Without weight windows, this is better than BFB. With the weight windows, we get an even smaller variance than with BFB with h raised to the power $18.2/14$, as reported earlier, but at the expense of a much longer simulation time, leading to poorer efficiency. For the case of a fixed number of chains, we were unable to obtain a result because it required too long simulation times. Thus here again, the approximation of $\gamma(x)$ by $h(x)$ needs to be improved. For instance, we have $h(x_0) = 1.4 \times 10^{-20}$ whereas $\gamma(x_0) = 4.2 \times 10^{-18}$. Similarly to what we did for BFB, we decided to raise the current h to the power $\alpha \approx 17.5/20$, to get a new h that matches $\gamma(x)$ at $x = x_0$. With this new h, the variance per run for weight windows is 6.6×10^{-35} with an average number of steps and average number of hits per chain of 458 and 0.49 (recall that for this example, we always take $n = 4$ and $m = 2^{20}$). Thus, the variance is slightly larger than with the initial h, but the simulation time is reduced by a factor 10. With a fixed number of chains, we obtained a variance per run of 8.0×10^{-34} and an average number of steps per chain of 29.5. In summary, with DFB, the weight windows are not very effective to improve the efficiency with this choice of h.

5. CONCLUSION

IS is one the most widely used variance reduction technique in rare event simulation, though choosing a proper IS measure is often a difficult task. In this paper, we have proposed to alleviate this problem by combining IS with a weight-window technique that kills the chains whose weighted importance is deemed too small and split those whose weighted importance appears too large, so that all the chains have a weighted importance around the same value. Our (very preliminary) numerical examples with highly reliable Markovian systems indicate that this can reduce substantially the variance of the estimator. One difficulty, on the other hand, is that the method requires a reasonably good approximation of the function γ.

This work is preliminary. As directions of future research, we aim at investigating a proper choice of importance function for specific classes of problems, and provide hints to ensure a variance reduction. We also want to explore the combination with multilevel splitting, which we have described but not tested empirically.

Acknowledgment

This research has been supported by NSERC-Canada grant No. ODGP0110050 and a Canada Research Chair to the first author, EuroNGI Network of Excellence, INRIA's cooperative research action RARE and "SurePath ACI Security" Project to the second author, and an FQRNT-INRIA Travel Grant to both authors. Part of the work was accomplished while the first author was enjoying the hospitality of IRISA, in Rennes.

6. REFERENCES

[1] T. E. Booth. Automatic importance estimation in forward Monte Carlo calculations. *Transactions of the American Nuclear Society*, 41:308–309, 1982.

[2] T. E. Booth and J. S. Hendricks. Importance estimation in forward Monte Carlo estimation. *Nuclear Technology/Fusion*, 5:90–100, 1984.

[3] J. A. Bucklew. *Introduction to Rare Event Simulation*. Springer-Verlag, New York, 2004.

[4] H. Cancela, G. Rubino, and B. Tuffin. MTTF estimation by Monte Carlo methods using Markov models. *Monte Carlo Methods and Applications*, 8(4):312–341, 2002.

[5] H. Cancela, G. Rubino, and B. Tuffin. New measures of robustness in rare event simulation. In M. E. Kuhl, N. M. Steiger, F. B. Armstrong, and J. A. Joines, editors, *Proceedings of the 2005 Winter Simulation Conference*, pages 519–527, 2005.

[6] J. A. Carrasco. Failure distance based on simulation of repairable fault tolerant systems. *Proceedings of the 5th International Conference on Modeling Techniques and Tools for Computer Performance Evaluation*, pages 351–365, 1992.

[7] P. Del Moral. *Feynman-Kac Formulae. Genealogical and Interacting Particle Systems with Applications*. Probability and its Applications. Springer, New York, 2004.

[8] B. L. Fox. *Strategies for Quasi-Monte Carlo*. Kluwer Academic, Boston, MA, 1999.

[9] M. J. J. Garvels. *The Splitting Method in Rare Event Simulation*. PhD thesis, Faculty of mathematical Science, University of Twente, The Netherlands, 2000.

[10] M. J. J. Garvels and D. P. Kroese. A comparison of RESTART implementations. In *Proceedings of the 1998 Winter Simulation Conference*, pages 601–609. IEEE Press, 1998.

[11] M. J. J. Garvels, D. P. Kroese, and J.-K. C. W. Van Ommeren. On the importance function in splitting simulation. *European Transactions on Telecommunications*, 13(4):363–371, 2002.

[12] P. Glasserman, P. Heidelberger, and P. Shahabuddin. Asymptotically optimal importance sampling and stratification for pricing path dependent options. *Mathematical Finance*, 9(2):117–152, 1999.

[13] P. Glasserman, P. Heidelberger, P. Shahabuddin, and T. Zajic. Multilevel splitting for estimating rare event probabilities. *Operations Research*, 47(4):585–600, 1999.

[14] P. W. Glynn. Efficiency improvement techniques. *Annals of Operations Research*, 53:175–197, 1994.

[15] P. W. Glynn and D. L. Iglehart. Importance sampling for stochastic simulations. *Management Science*, 35:1367–1392, 1989.

[16] A. Goyal, P. Shahabuddin, P. Heidelberger, V. F. Nicola, and P. W. Glynn. A unified framework for simulating markovian models of highly reliable systems. *IEEE Transactions on Computers*, C-41:36–51, 1992.

[17] P. Heidelberger. Fast simulation of rare events in queueing and reliability models. *ACM Transactions on Modeling and Computer Simulation*, 5(1):43–85, 1995.

[18] H. Kahn and T. E. Harris. Estimation of particle transmission by random sampling. *National Bureau of Standards Applied Mathematical Series*, 12:27–30, 1951.

[19] V. B. Melas. On the efficiency of the splitting and roulette approach for sensitivity analysis. In *Proceedings of the 1997 Winter Simulation Conference*, pages 269–274, Piscataway, NJ, 1997. IEEE Press.

[20] M. K. Nakayama. General conditions for bounded relative error in simulations of highly reliable Markovian systems. *Advances in Applied Probability*, 28:687–727, 1996.

[21] P. Shahabuddin. Fast transient simulation of Markovian models of highly dependable systems. *Performance Evaluation*, 20:267–286, 1994.

[22] P. Shahabuddin. Importance sampling for the simulation of highly reliable markovian systems. *Management Science*, 40(3):333–352, 1994.

[23] B. Tuffin. Bounded normal approximation in simulations of highly reliable Markovian systems. *Journal of Applied Probability*, 36(4):974–986, 1999.

[24] B. Tuffin. On numerical problems in simulations of highly reliable Markovian systems. In *Proceedings of the 1st International Conference on Quantitative Evaluation of SysTems (QEST)*, pages 156–164, University of Twente, Enschede, The Netherlands, September 2004. IEEE CS Press.

[25] M. Villén-Altamirano and J. Villén-Altamirano. On the efficiency of RESTART for multidimensional systems. manuscript, 2006.