Rare Events, Splitting, and Quasi-Monte Carlo

PIERRE L'ECUYER and VALÉRIE DEMERS
Université de Montréal, Canada
and
BRUNO TUFFIN
IRISA–INRIA, Rennes, France

In the context of rare-event simulation, splitting and importance sampling (IS) are the primary approaches to make important rare events happen more frequently in a simulation and yet recover an unbiased estimator of the target performance measure, with much smaller variance than a straightforward Monte Carlo (MC) estimator. Randomized quasi-Monte Carlo (RQMC) is another class of methods for reducing the noise of simulation estimators, by sampling more evenly than with standard MC. It typically works well for simulations that depend mostly on very few random numbers. In splitting and IS, on the other hand, we often simulate Markov chains whose sample paths are a function of a long sequence of independent random numbers generated during the simulation. In this paper, we show that RQMC can be used jointly with splitting and/or IS to construct better estimators than those obtained by either of these methods alone. We do that in a setting where the goal is to estimate the probability of reaching B before reaching (or returning to) A when starting from a distinguished state not in B, where A and B are two disjoint subsets of the state space and B is very rarely reached. This problem has several practical applications. The paper is in fact a two-in-one: the first part provides a guided tour of splitting techniques, introducing along the way some improvements in the implementation of the multilevel splitting. At the end of the paper, we also give examples of situations where splitting is not effective. For these examples, we compare different ways of applying IS and combining it with RQMC.

Categories and Subject Descriptors: G.3 [**Mathematics of Computing**]: Probability and Statistics; I.6 [**Computing Methodology**]: Simulation and Modeling

Additional Key Words and Phrases: Quasi-Monte Carlo, Markov chain, variance reduction, splitting, RESTART, importance sampling, highly-reliable Markovian systems

1. INTRODUCTION

Rare-event simulation was the prime focus of Perwez Shahabuddin's research. He gave us crisp insight on several aspects of its two main tools: importance sampling (IS) and splitting. For IS, he worked on both the light-tail and the (more difficult)

Authors' addresses: Pierre L'Ecuyer and Valérie Demers, Département d'Informatique et de Recherche Opérationnelle, Université de Montréal, C.P. 6128, Succ. Centre-Ville, Montréal, H3C 3J7, Canada, e-mail: lecuyer@iro.umontreal.ca, demersv@iro.umontreal.ca; Bruno Tuffin, IRISA-INRIA, Campus Universitaire de Beaulieu, 35042 Rennes Cedex, France, e-mail: Bruno.Tuffin@irisa.fr.

heavy-tail settings. He also developed clever simulation techniques for Markov chains representing various classes of highly-reliable systems. For splitting, he and his co-authors used branching-process theory to find the optimal degree of splitting, gave conditions under which the method is asymptotically efficient, and shed light on (potential) practical difficulties with the method. In all these cases, he came up with sharp theorems, most of them on the asymptotic behavior as the probability or the frequency of the important events goes to zero. His contributions cover the major application areas of rare-event simulation in our community: reliability, communication networks (e.g., buffer overflow probabilities), finance, and insurance risk, for example.

In this paper, we build on part of his work by examining how IS and splitting can be combined with randomized quasi-Monte Carlo (RQMC) to improve their efficiency even further. The setting is a discrete-time Markov chain simulated until a random stopping time. We consider two types of RQMC approaches. The first one (called classical RQMC) takes a high-dimensional RQMC point set and uses each point to simulate an entire sample path of the system. The other one (called array-RQMC) simulates several copies of the chain in parallel and uses a different low-dimensional RQMC point set at each step to move all the chains to the next step. We also introduce unbiased variants of splitting that trade a small variance increase for a more significant work reduction, and fit well with the array-RQMC approach.

Our backbone model is a discrete-time Markov chain $\{X_j,\, j \geq 0\}$ with arbitrary state space $\mathcal{X}$. Let A and B be two disjoint subsets of $\mathcal{X}$ and let $x_0 \in \mathcal{X} \setminus B$, the initial state. Often, $x_0 \in A$. The chain starts in state $X_0 = x_0$, leaves the set A if $x_0 \in A$, and then eventually reaches B or A. Let $\tau_A = \inf\{j > 0 : X_{j-1} \notin A \text{ and } X_j \in A\}$, the first time when the chain hits A (or returns to A after leaving it), and $\tau_B = \inf\{j > 0 : X_j \in B\}$, the first time when the chain reaches the set B. The goal is to estimate $\gamma_0 = \mathbb{P}[\tau_B < \tau_A]$, the probability that the chain reaches B before A. This particular form of rare-event problem, where γ_0 is small, occurs in many practical situations; see, e.g., Shahabuddin et al. [1988], Shahabuddin [1994b], Shahabuddin [1994a], and Heidelberger [1995].

To estimate γ_0, the *standard Monte Carlo* method runs n independent copies of the chain up to the stopping time $\tau = \min[\tau_A, \tau_B]$, counts how many times the event $\{\tau_B < \tau_A\}$ occurs, and divides by n. The resulting estimator $\hat{\gamma}_n$ is highly unreliable (almost useless) when the probability γ_0 is very small. For example, if $\gamma_0 = 10^{-10}$ and if we want the expected number of occurrences of this event to be at least 100, we must take $n = 10^{12}$ (a huge number). For $n < 10^{10}$, we are likely to observe *not even a single occurrence* of this event. In this case, not only the estimator of γ_0 takes the value 0 but the empirical variance as well, which can be quite misleading if we use the empirical mean and variance to compute a confidence interval on γ_0. The estimator $\hat{\gamma}_n$ has the binomial distribution with parameters (n, γ_0). Its *relative error* is

$$\mathrm{RE}[\hat{\gamma}_n] = \frac{(\mathrm{Var}[\hat{\gamma}_n])^{1/2}}{\gamma_0} = \frac{(\gamma_0(1-\gamma_0)/n)^{1/2}}{\gamma_0} \approx (\gamma_0 n)^{-1/2},$$

which increases toward infinity when $\gamma_0 \to 0$. An alternative unbiased estimator of γ_0, say $\tilde{\gamma}_n$, is said to have *bounded relative error* if $\lim_{\gamma_0 \to 0^+} \mathrm{RE}[\tilde{\gamma}_n] < \infty$. This

implies that

$$\lim_{\gamma_0 \to 0^+} \frac{\log(\mathbb{E}[\tilde{\gamma}_n^2])}{\log \gamma_0} = 2. \tag{1}$$

When the latter (weaker) condition holds, the estimator $\tilde{\gamma}_n$ is said to be *asymptotically efficient* [Heidelberger 1995; Bucklew 2004]. To take into account the computing cost of the estimator, it is common practice to consider the *efficiency* of an estimator $\tilde{\gamma}_n$ of γ_0, defined as $\mathrm{Eff}[\tilde{\gamma}_n] = 1/(\mathrm{Var}[\tilde{\gamma}_n]C(\tilde{\gamma}_n))$ where $C(\tilde{\gamma}_n)$ is the expected time to compute $\tilde{\gamma}_n$. *Efficiency improvement* means finding an unbiased estimator with larger efficiency than the one previously available. The *work-normalized relative error* of $\tilde{\gamma}_n$ is defined as $\mathrm{RE}[\hat{\gamma}_n][C(\tilde{\gamma}_n)]^{1/2}$. We say that $\tilde{\gamma}_n$ has *bounded work-normalized relative error* if $\lim_{\gamma_0 \to 0^+} \gamma_0^2 \mathrm{Eff}[\tilde{\gamma}_n] > 0$. It is *work-normalized asymptotically efficient* (a weaker condition) if $\lim_{\gamma_0 \to 0^+} \log(C(\tilde{\gamma}_n)\mathbb{E}[\tilde{\gamma}_n^2])/\log \gamma_0 = 2$. A sufficient condition for this is that (1) holds and $\lim_{\gamma_0 \to 0^+} \log C(\tilde{\gamma}_n)/\log \gamma_0 = 0$.

Example 1. Following Shahabuddin [1994b], we consider a highly-reliable Markovian system (HRMS) with c types of components and n_i components of type i, for $i = 1, \ldots, c$. We assume that the system's evolution can be represented by a *continuous-time Markov chain* (CTMC) $\{Y(t) = (Y_1(t), \ldots, Y_c(t)),\, t \geq 0\}$, where $Y_i(t) \in \{0, \ldots, n_i\}$ is the number of failed components of type i at time t. This CTMC has a finite state space of cardinality $(n_1+1)\cdots(n_c+1)$, which can be huge when c is large.

When the state $Y(t)$ satisfies certain conditions, the system is up (operational), otherwise it is down (failed). The set B here is the set of states in which the system is down. Typically, this is an increasing set: if $y \in B$ and $y' \geq y$, then $y' \in B$. Let $A = \{x_0\}$ be the set that contains only the state $x_0 = (0, \ldots, 0)$ in which all components are operational. Each transition of the CTMC corresponds to either the failure of an operational component or the repair of a failed component. Failure and repair rates may depend on the entire state.

Let $\xi_0 = 0$, let $0 \leq \xi_1 \leq \xi_2 \leq \cdots$ be the jump times of this CTMC, and let $\{X_j,\, j \geq 0\}$ be its discrete skeleton (or embedded discrete-time Markov chain), defined by $X_j = Y(\xi_j)$ for $j \geq 0$. We also define $\Delta_j = \mathbb{E}[\xi_{j+1} - \xi_j \mid Y(\xi_j)]$, the expected sojourn time in state X_j, which equals the inverse of the jump rate out of that state. The time until the first system failure is

$$\tilde{\tau}_B = \sum_{j=0}^{\tau_B - 1} (\xi_{j+1} - \xi_j).$$

Suppose we want to estimate the *mean time to failure* (MTTF) from state x_0, defined as

$$\mathbb{E}[\tilde{\tau}_B] = \mathbb{E}\left[\sum_{j=0}^{\tau_B - 1} \Delta_j\right].$$

Each return to state x_0 is a regeneration point for this system. So, if we define $\tau = \min(\tau_A, \tau_B)$ and $\tilde{\tau} = \sum_{j=0}^{\tau-1} \Delta_j$, a standard renewal argument (e.g., Keilson 1979) gives

$$\mathbb{E}[\tilde{\tau}_B] \;=\; \mathbb{E}[\tilde{\tau}] + \mathbb{E}[\tilde{\tau}_B - \tilde{\tau} | \tau_A < \tau_B]\mathbb{P}[\tau_A < \tau_B]$$

$$\begin{aligned} &= \mathbb{E}[\tilde{\tau}] + \mathbb{E}[\tilde{\tau}_B - \tilde{\tau} | X_\tau = X_{\tau_A} = x_0]\mathbb{P}[\tau_A < \tau_B] \\ &= \mathbb{E}[\tilde{\tau}] + \mathbb{E}[\tilde{\tau}_B](1 - \gamma_0) \end{aligned}$$

and then

$$\mathbb{E}[\tilde{\tau}_B] = \mathbb{E}[\tilde{\tau}]/\gamma_0.$$

If the system returns frequently to x_0 and rarely reaches failure, $\mathbb{E}[\tilde{\tau}]$ is usually easy to estimate by standard simulation, but γ_0 is difficult to estimate. Shahabuddin et al. [1988] recommend to devote 10% of the computing budget to estimate the numerator by standard Monte Carlo and 90% to estimate the denominator with IS, using separate simulations. Shahabuddin [1994b] developed efficient IS schemes for estimating γ_0 in this context and gave conditions under which the estimator has bounded relative error when $\gamma_0 \to 0$. Simulation of HRMS was also studied by Goyal et al. [1992], Nicola et al. [1993], Shahabuddin [1994a], Heidelberger et al. [1994], Nakayama [1996], Nakayama and Shahabuddin [1998], Juneja and Shahabuddin [2001], and Cancela et al. [2002], among others.

HRMS are used for instance to model fault-tolerant computer systems and design the configuration of their components (disks, processors, etc.) to make them highly dependable. Typical measures of interest include the MTTF, the steady-state availability, and the reliability (the probability that the system does not fail before a given time horizon). Even with the assumption of exponential laws for failures and repairs of components, analytical or numerical solution methods are impractical due to the excessive number of states of the CTMC, and simulation becomes the only viable technique of analysis.

Example 2. A similar problem occurs in a queueing system when we want to estimate the expected time until the number of customers in a queue exceeds a given value [Parekh and Walrand 1989; Chang et al. 1994; Heidelberger 1995; Shahabuddin 1995]. For example, the customers could be packets in a telecommunication network, where the queue represents the packets stored in a buffer at a given node (or switch) and waiting for their turn to be transmitted, and the number to exceed is the size of the buffer. The set A contains the single state where the system is empty, B represents the states for which the buffer overflows, γ_0 is the probability that the buffer overflows before returning to empty, and $\mathbb{E}[\tilde{\tau}_B]$ is the expected time to overflow.

IS and splitting are the two primary techniques to deal with rare-event simulation. IS changes the probability laws that drive the evolution of the system, to increase the probability of the rare event, and multiplies the estimator by an appropriate likelihood ratio so that it has the correct expectation (e.g., remains unbiased for γ_0 in the above setting). A major difficulty in general is to find a good way to change the probability laws. For the details, we refer the reader to Glynn and Iglehart [1989], Glynn [1994], Heidelberger [1995], Andradóttir et al. [1995], Asmussen [2002], Bucklew [2004], and many other references given there.

In the splitting method, the probability laws of the system remain unchanged, but an artificial drift toward the rare event is created by terminating the trajectories that seem to go away from it and by *splitting* (cloning) those that are going in the right direction. In some settings, an unbiased estimator is recovered by multiplying

the original estimator by an appropriate factor. We give more details in the next section. The method can be traced back to Kahn and Harris [1951] and has been studied (sometimes under different names) by several authors, including Booth and Hendricks [1984], Villén-Altamirano and Villén-Altamirano [1994], Melas [1997], Garvels and Kroese [1998], Glasserman et al. [1998], Glasserman et al. [1999], Fox [1999], Garvels [2000], Del Moral [2004], Cérou et al. [2005], Villén-Altamirano and Villén-Altamirano [2006], and other references cited there.

In this paper, we concentrate mainly on the splitting method and examine how it can be combined with *randomized quasi-Monte Carlo* (RQMC) to further reduce the variance. The *array-RQMC* method recently proposed by L'Ecuyer et al. [2005], appears (at first sight) highly compatible with splitting, because both techniques work with an array of Markov chains simulated in parallel and are designed primarily for Markov chains that evolve for a large number of steps. Our aim is to examine if and how they can be combined and assess the degree of improvement obtained by their combination, as well as the difficulties that must be tackled to obtain this additional gain. Along the way, we propose some improvements in the implementation of the multilevel splitting algorithm. We also illustrate how IS can be combined with RQMC and array-RQMC for some specific applications where splitting is not relevant.

The remainder of the paper is organized as follows. In the next section, we review the theory and practice of splitting in the setting where we want to estimate $\gamma_0 = \mathbb{P}[\tau_B < \tau_A]$. We propose new variants, more efficient than the standard implementations. In Section 3, we describe the RQMC and array-RQMC methods and how they can be implemented in our setting. In Section 4, we discuss the potential difficulties of the array-RQMC method in this context. Numerical illustrations are given in Section 5 with several examples. We start with an Ornstein-Uhlenbeck (mean-reverting) process for which B is the set of states that exceed a given threshold. The second example is a tandem queue where B is the set of states where the number of customers waiting at the second queue exceeds a given value. Then we consider examples of highly-reliable Markovian systems similar to those examined by Shahabuddin [1994b] and Shahabuddin [1994a]. A conclusion summarizes our findings.

2. SPLITTING

2.1 Multilevel Splitting

It is customary to define the multilevel splitting algorithm via an *importance function* $h : \mathcal{X} \to \mathbb{R}$ that assigns a real number to each state of the chain [Garvels et al. 2002]. We shall assume that $A = \{x \in \mathcal{X} : h(x) \leq 0\}$ and $B = \{x \in \mathcal{X} : h(x) \geq \ell\}$ for some constant $\ell > 0$. In the multilevel splitting method, we partition the interval $[0, \ell)$ in m subintervals with boundaries $0 = \ell_0 < \ell_1 < \cdots < \ell_m = \ell$. For $k = 1, \ldots, m$, define $T_k = \inf\{j > 0 : h(X_j) \geq \ell_k\}$, let $D_k = \{T_k < \tau_A\}$ denote the event that $h(X_j)$ reaches level ℓ_k before reaching level 0, and define the conditional probabilities $p_k = \mathbb{P}[D_k \mid D_{k-1}]$ for $k > 1$, and $p_1 = \mathbb{P}[D_1]$. Since

$D_m \subset D_{m-1} \subset \cdots \subset D_1$, we see immediately that

$$\gamma_0 = \mathbb{P}[D_m] = \prod_{k=1}^{m} p_k. \tag{2}$$

The basic idea of multilevel splitting is to estimate each probability p_k "separately", by starting a large number of chains in states that are generated from the distribution of $X_{T_{k-1}}$ conditional on the event D_{k-1}. This conditional distribution, denoted by G_{k-1}, is called the (first-time) *entrance distribution at threshold* ℓ_{k-1}, for $k = 1, \ldots, m+1$ (G_0 is degenerate at x_0). Conceptually, this estimation is done in successive *stages*, as follows.

In the first stage, we start N_0 independent chains from the initial state x_0 and simulate each of them until time $\min(\tau_A, T_1)$. Let R_1 be the number of those chains for which D_1 occurs. Then $\hat{p}_1 = R_1/N_0$ is an obvious unbiased estimator of p_1. The empirical distribution $\hat{G}_1$ of these R_1 entrance states X_{T_1} can be viewed as an estimate of the conditional distribution G_1.

In the second stage, if $R_1 > 0$, we start N_1 chains from these R_1 entrance states, by cloning (splitting) some chains if we want $N_1 > R_1$, and continue the simulation of these chains independently up to time $\min(\tau_A, T_2)$. Note that the initial state of each of these N_1 chains at the beginning of the second stage has distribution G_1, so for each of these chains, the event D_2 has probability p_2 and the entrance state at the next level if D_2 occurs has distribution G_2. Then $\hat{p}_2 = R_2/N_1$ is an unbiased estimator of p_2, where R_2 is the number of those chains for which D_2 occurs. If $R_1 = 0$, then $\hat{p}_k = 0$ for all $k \geq 1$ and the algorithm stops here.

This procedure is repeated at each stage. In stage k, ideally we would like to generate N_{k-1} states independently from the entrance distribution G_{k-1}. Or even better, to generate a stratified sample from G_{k-1}. But we usually cannot do that, because G_{k-1} is unknown. Instead, we pick N_{k-1} states out of the R_{k-1} that are available (by cloning if necessary), simulate independently from these states up to time $\min(\tau_A, T_k)$, and estimate p_k by $\hat{p}_k = R_k/N_{k-1}$ where R_k is the number of chains for which D_k occurs. If $R_k = 0$, we put $\hat{p}_j = 0$ for all $j \geq k$ and the algorithm simply returns $\hat{\gamma}_n = 0$ right away (this could occur with non-negligible probability if $p_k N_k$ is not large enough.) The initial state of each of the N_{k-1} chains at the beginning of stage k has (unconditional) distribution G_{k-1}. Thus, for each of these chains, the event D_k has probability p_k and the entrance state at the next level if D_k occurs has distribution G_k.

Even though the $\hat{p}_k$'s are not independent, it is easy to prove by induction on k that the product $\hat{p}_1 \cdots \hat{p}_m = (R_1/N_0)(R_2/N_1)\cdots(R_m/N_{m-1})$ is an unbiased estimator of γ_0 [Garvels 2000, page 17]: If we assume that $\mathbb{E}[\hat{p}_1 \cdots \hat{p}_{k-1}] = p_1 \cdots p_{k-1}$, then

$$\begin{aligned}
\mathbb{E}[\hat{p}_1 \cdots \hat{p}_k] &= \mathbb{E}[\hat{p}_1 \cdots \hat{p}_{k-1} \mathbb{E}[\hat{p}_k \mid N_0, \ldots, N_{k-1}, R_1, \ldots, R_{k-1}]] \\
&= \mathbb{E}[\hat{p}_1 \cdots \hat{p}_{k-1}(N_{k-1}p_k)/N_{k-1}] \\
&= p_1 \cdots p_k.
\end{aligned} \tag{3}$$

Combining this with the fact that $\mathbb{E}[\hat{p}_1] = p_1$, the result follows. Splitting can improve the efficiency by increasing the number of chains that reach the rare set B.

2.2 Fixed splitting vs fixed effort

There are many ways of doing the splitting [Garvels 2000]. For example, one may clone each of the R_k chains that reached level k in c_k copies, for a fixed positive integer c_k. Then, each $N_k = c_k R_k$ is random. This is *fixed splitting*. Often, we want the *expected* number of splits of each chain to be c_k, where c_k is not necessarily an integer; say $c_k = \lfloor c_k \rfloor + \delta$ where $0 \le \delta < 1$. In this case, we assume that the actual number of splits is $\lfloor c_k \rfloor + 1$ with probability δ and $\lfloor c_k \rfloor$ with probability $1 - \delta$.

In the *fixed effort* method, in contrast, we fix each value of N_k a priori and make just the right amount of splitting to reach this target value. This can be achieved by *random assignment*: draw the N_k starting states at random, with replacement, from the R_k available states. This is equivalent to sampling from the empirical distribution $\hat{G}_k$ of these R_k states. In a *fixed assignment*, on the other hand, we would split each of the R_k states approximately the same number of times as follows. Let $c_k = \lfloor N_k / R_k \rfloor$ and $d_k = N_k \bmod R_k$. Select d_k of the R_k states at random, without replacement. Each selected state is split $c_k + 1$ times and the other states are split c_k times. The fixed assignment gives a smaller variance than the random assignment because it amounts to using stratified sampling over the empirical distribution G_k at level k.

These variants are all unbiased, but they differ in terms of variance. Garvels and Kroese [1998] conclude from their analysis and empirical experiments that fixed effort performs better, mainly because it reduces the variance of the number of chains that are simulated at each stage. In the next subsection, we show that with optimal splitting factors, this is not always true.

The probability γ_0 can also be seen as a normalization constant in Feynman-Kac formulas. To estimate this constant, Del Moral [2004] proposes and studies approximation algorithms based on interacting particle systems that exploits decomposition (2). These algorithms are essentially equivalent to the presently described fixed-effort implementation with random assignment. In this type of system, particles that did not reach the threshold are killed and replaced by clones of randomly selected particles among those that have succeeded. This redistributes the effort on most promising particles while keeping the total number constant. Cérou et al. [2005] derive limit theorems for the corresponding estimators.

2.3 Variance analysis for a simplified setting

For a very crude variance analysis, consider an idealized fixed-effort setting where

$$N_0 = N_1 = \cdots = N_{m-1} = n \tag{4}$$

and where the $\hat{p}_k$'s are independent binomial random variables with parameters n and $p_k = p = \gamma_0^{1/m}$. Then, for $m > 1$,

$$\begin{aligned}
\mathrm{Var}[\hat{p}_1 \cdots \hat{p}_m] &= \prod_{k=1}^{m} \mathbb{E}[\hat{p}_k^2] - \gamma_0^2 \;=\; \prod_{k=1}^{m} \left(p_k^2 + \frac{p_k(1-p_k)}{n} \right) - \gamma_0^2 \\
&= \left(p^2 + \frac{p(1-p)}{n} \right)^m - p^{2m} \\
&= \frac{m p^{2m-1}(1-p)}{n} + \frac{m(m-1)p^{2m-2}(1-p)^2}{2n^2} \;+\; \cdots \;+ \frac{(p(1-p))^m}{n^m}.
\end{aligned}$$

If we assume that

$$n \gg (m-1)(1-p)/p, \tag{5}$$

the dominant term in the last expression is the first one, $mp^{2m-1}(1-p)/n \approx m\gamma_0^{2-1/m}/n$. The MC variance, on the other hand, is $\gamma_0(1-\gamma_0)/n \approx \gamma_0/n$. To illustrate the huge potential variance reduction, suppose $\gamma_0 = 10^{-20}$, $m = 20$, $p = 1/10$, and $n = 1000$. Then the MC variance is 10^{-23} whereas $mp^{2m-1}(1-p)/n \approx 1.8 \times 10^{-41}$. This oversimplified setting is not realistic, because the $\hat{p}_k$ are generally not independent and it is difficult to have $p_k = \gamma_0^{1/m}$ for all i. But it nevertheless gives an idea of the order of magnitude of potential variance reduction that can be achieved.

The amount of work (or CPU time, or number of steps simulated) at each stage is proportional to n, so the total work is proportional to nm. Most of this work is to simulate the n chains down to level 0 at each stage. Thus, the efficiency of the splitting estimator under the simplified setting is approximately proportional to $n/[\gamma_0^{2-1/m} nm^2] = \gamma_0^{-2+1/m}/m^2$ when (5) holds. By differentiating with respect to m, we find that this expression is maximized by taking $m = -\ln(\gamma_0)/2$ (we neglect the fact that m must be an integer). This gives $p^m = \gamma_0 = e^{-2m}$, so $p = e^{-2}$. Garvels and Kroese [1998] have obtained this result. The squared relative error in this case is (approximately) $\gamma_0^{2-1/m}(m/n)\gamma_0^{-2} = e^2 m/n = -e^2 \ln(\gamma_0)/(2n)$ and the work-normalized relative error is $m\gamma_0^{-1/(2m)} = -(e/2)\ln(\gamma_0)$, again under the condition (5).

When $\gamma_0 \to 0$ for fixed p, we have $m \to \infty$, so (5) does not hold. Then, the relative error and its work-normalized version both increase toward infinity at a logarithmic rate. This agrees with Garvels [2000, page 20]. With $\tilde{\gamma}_n = \hat{p}_1 \cdots \hat{p}_m$, the limit in (1) is

$$\lim_{\gamma_0 \to 0^+} \frac{\log(p^2 + p(1-p)/n)^m}{\log \gamma_0} = \lim_{\gamma_0 \to 0^+} \frac{-\log(p^2 + p(1-p)/n)}{-\log p} < 2.$$

Thus, this splitting estimator is not quite asymptotically efficient, but almost (when n is very large).

Consider now a fixed-splitting setting, assuming that $N_0 = n$, $p_k = p = \gamma_0^{1/m}$ for all k, and that the constant splitting factor at each stage is $c = 1/p$; i.e., $N_k = R_k/p$. Then, the process $\{N_k,\, k \geq 1\}$ is a *branching process* and the estimator becomes

$$\hat{p}_1 \cdots \hat{p}_m = \frac{R_1}{N_0}\frac{R_2}{N_1} \cdots \frac{R_m}{N_{m-1}} = \frac{R_m p^{m-1}}{n}.$$

From standard branching process theory [Harris 1963], we have that

$$\mathrm{Var}[\hat{p}_1 \cdots \hat{p}_m] = m(1-p)p^{2m-1}/n.$$

If p is fixed and $m \to \infty$, then the squared relative error $m(1-p)/(np)$ is unbounded here as well. However, the limit in (1) becomes

$$\lim_{\gamma_0 \to 0^+} \frac{\log(m(1-p)\gamma_0^2/(np) + \gamma_0^2)}{\log \gamma_0} = \lim_{\gamma_0 \to 0^+} \frac{-2m\log p - \log(1 + m(1-p)/(np))}{-m \log p} = 2,$$

so the splitting estimator is asymptotically efficient [Glasserman et al. 1999]. This implies that fixed splitting is asymptotically better in this case.

Glasserman et al. [1999] study the *fixed splitting* framework with splitting factor $c_k \equiv c$, for a countable-state space Markov chain. They assume that the probability transition matrix $\mathbf{P}_k$ for the first-entrance state at level k given the first-entrance state at level $k-1$ converges to a matrix $\mathbf{P}$ with spectral radius $\rho < 1$. This implies that $p_k \to \rho$ when $k \to \infty$. Then they use branching process theory to prove that the multilevel splitting estimator (in their setting) is work-normalized asymptotically efficient if and only if $c = 1/\rho$. Glasserman et al. [1998] show that the condition $c = 1/\rho$ is not sufficient for asymptotic efficiency and provide additional necessary conditions in a general multidimensional setting. Their results highlight the crucial importance of choosing a good importance function h.

Even though fixed splitting is asymptotically better under ideal conditions, its efficiency is extremely sensitive to the choice of splitting factors. If the splitting factors are too high, the number of chains (and the amount of work) explodes, whereas if they are too low, the variance is very large because very few chains reach B. Since the optimal splitting factors are unknown in real-life applications, the more robust fixed-effort approach is usually preferable. Fixed effort also has better compatibility with RQMC than fixed splitting (see Section 3).

2.4 Implementation issues

One drawback of the fixed-effort approach is that it requires more memory than the fixed splitting version, because it must use a *breadth-first* implementation: at each stage k all the chains must be simulated until they reach either A or level ℓ_k before we know the splitting factor at that level. The states of all the chains that reach ℓ_k must be saved; this may require too much memory when the N_k's are large. With fixed splitting, we can adopt a *depth-first* strategy, where each chain is simulated entirely until it hits ℓ or A, then its most recent clones (created at the highest level that it has reached) are simulated entirely, then those at the next highest level, and so on. This procedure is applied recursively. At most one state per level must be memorized with this approach. This is feasible because the amount of splitting at each level is fixed a priori.

An important part of the work in multilevel splitting is due to the fact that all the chains considered in stage k (from level ℓ_{k-1}) and which do not reach ℓ_k must be simulated until they get down to A. When ℓ_{k-1} is large, this can take significant time. Because of this, the expected amount of work increases with the number of thresholds. One (heuristic) way of reducing this work in exchange for a small bias is to truncate the chains that reach level $\ell_{k-\beta}$ downward after they have reached ℓ_{k-1}, for some fixed integer $\beta \geq 2$, and assume that all these chains will go down to A. The integer β must be taken large enough so that the probability that a chain starting at level $\ell_{k-\beta}$ will get back up to ℓ_k is very small. The larger this probability, the larger the bias of the final estimator. In Section 2.7, we discuss unbiased alternatives, based on variants of the Russian roulette principle.

2.5 The RESTART Algorithm

The *RESTART* method [Villén-Altamirano and Villén-Altamirano 1994; 2006] is a variant of splitting where any chain is split by a fixed factor when it hits a level upward, and one of the copies is tagged as the *original* for that level. When any of those copies hits that same level downward, if it is the original it just continues

its path, otherwise it is killed immediately. This rule applies recursively, and the method is implemented in a depth-first fashion, as follows: whenever there is a split, all the copies are simulated completely, one after the other, then simulation continues for the original chain. Unbiasedness is proved by Garvels [2000] and Villén-Altamirano and Villén-Altamirano [2002]. The reason for killing most of the paths that go downward is to reduce the work. The number of paths that are simulated down to A never exceeds N_0. On the other hand, the number of chains that reach a given level is more variable with this method than with the fixed-effort and fixed-assignment multilevel splitting algorithm described previously. As a result, the final estimator of γ_0 has a larger variance [Garvels 2000]. Another source of additional variance is that the resplits tend to share a longer common history and to be more positively correlated. This source of variance can be important when the probability of reaching B from a given level varies significantly with the entrance state at that level [Garvels 2000]. In terms of overall efficiency, none of the two methods is universally better; RESTART wins in some cases and splitting wins in other cases. Villén-Altamirano and Villén-Altamirano [2002] provide a detailed variance analysis of RESTART.

The RESTART algorithm can also be implemented with a *breadth first* approach, by advancing all the chains by one step at every step of the algorithm. Each chain has a tag that indicates its level of creation. For the original chains that started at level 0, this tag indicates 0, otherwise it is the level where it was initiated by a split. When a chain goes down to its level of creation, it is killed. Whenever a chain reaches a level from below, it is split and the new clones are tagged with this level. The algorithm stops when all the chains have been killed or have reached B.

2.6 Choice of the Importance Function and Optimal Parameters

Key issues in multilevel splitting are the choices of the importance function h, levels ℓ_k, and splitting factors. To discuss this, we introduce some more notation. Let $\mathcal{X}_k \subset \mathcal{X}$ be the support of the entrance distribution G_k, i.e., the states in which the chain can possibly be when hitting level ℓ_k for the first time. Let

$$\gamma(x) = \mathbb{P}[\tau_B < \tau_A \mid \tau > j,\, X_j = x],$$

the probability of reaching B before A if the chain is currently in state x, and $p_k(x) = \mathbb{P}[D_k \mid D_{k-1}, X_{T_{k-1}} = x]$, the probability of reaching level k before hitting A if the chain has just entered level $k-1$ in state x, for $x \in \mathcal{X}_{k-1}$. Note that $p_k = \int_{x \in \mathcal{X}_{k-1}} p_k(x) dG_{k-1}(x)$ and $\gamma_0 = \gamma(x_0)$.

One-dimensional case: selecting the levels. If the Markov chain has a *one-dimensional state space* $\mathcal{X} \subset \mathbb{R}$, $\gamma(x)$ is increasing in x, and if $A = (-\infty, 0]$ and $B = [\ell, \infty)$ for some constant ℓ, then we could simply choose $h(x) = x$ (or any strictly increasing function). In this case, the kth level is attained when the *state* reaches the value ℓ_k. This value need not be reached exactly: in general, the chain can jump directly from a smaller value to a value larger than ℓ_k, perhaps even larger than ℓ_{k+1}. So even in the one-dimensional case, the entrance state x at a given level is not unique in general and the probability $p_k(x)$ of reaching the next level depends on this (random) entrance state. It remains to choose the levels ℓ_k.

We saw earlier that in a fixed effort setting and under simplifying assumptions,

it is optimal to have $p_k \equiv p = e^{-2}$ for all k. This gives $m = -\ln(\gamma_0)/2$ levels. Note that to obtain equal p_k's, it is typically necessary to take unequal distances between the successive levels ℓ_k, i.e., $\ell_k - \ell_{k-1}$ must depend on k.

Suppose now that we use fixed splitting with $c_k = 1/p_k = e^2$ for each k. If we assume (very crudely) that each chain is split by a factor of e^2 at each stage, the total number of copies of a single initial chain that have a chance to reach B is

$$e^{2m-2} = e^{-\ln(\gamma_0)-2} = e^{-2}\gamma_0^{-1}. \tag{6}$$

Since each one reaches B with probability γ_0, this crude argument indicates that the expected number of chains that reach B is approximately equal to $p = e^{-2}$ times the initial number of chains at stage 0, exactly as in the fixed-effort case. However, the *variance* generally differs.

Cérou and Guyader [2005] determine the thresholds adaptively for the splitting with fixed effort in dimension 1. They first simulate n chains (trajectories) until these chains reach A or B. Then they sort the chains according to the maximum value of the importance function h that each chain has reached. The k trajectories with the largest values are kept, while the $n-k$ others are re-simulated, starting from the state at which the highest value of the importance function was obtained for the $(n-k)$-th largest ones. They proceed like this until $n-k$ trajectories have reached B. Their estimator is proved to be consistent, but is biased.

Multidimensional case: defining the importance function. In the case of a multidimensional state space, the choice of h is much more difficult. Note that h and the ℓ_k's jointly determine the probabilities $p_k(x)$ and p_k. Garvels et al. [2002] show by induction on k that for any fixed $p_1, \ldots, p_m$, h should be defined so that $p_k(x) = p_k$ (independent of x) for all $x \in \mathcal{X}_{k-1}$ and all k. This rule minimizes the residual variance of the estimator from stage k onward. With an h that satisfies this condition, the optimal levels and splitting factors are the same as in the one-dimensional case: $m = -(1/2)\ln\gamma_0$ levels, $p_k \approx e^{-2}$ and $\mathbb{E}[N_k] = N_0$ for each k. A simple choice of h and ℓ_k's that satisfies these conditions is

$$h(x) = h^*(x) \stackrel{\text{def}}{=} \gamma(x) \quad \text{and} \quad \ell_k = e^{-2(m-k)} = \gamma_0 e^{2k}.$$

Garvels et al. [2002] gave the following (equivalent) alternative choice: $\ell_k = k$ for each k and

$$h(x) \;=\; h^{**}(x) \;\stackrel{\text{def}}{=}\; \frac{\ln(\gamma(x)/\gamma_0)}{2} = m + \frac{\ln(\gamma(x))}{2}$$

for all $x \in \mathcal{X}$. However, these levels are optimal only if we assume that the chain can reach ℓ_k only on the set $\{x : \gamma(x) = e^{-2(m-k)}\}$, an optimistic assumption that rarely holds in practice, especially in the multidimensional case.

Garvels et al. [2002] also show how to get a first estimate of $\gamma(x)$ beforehand, in simple situations where the Markov chain has a finite state space, by simulating the chain backward in time. They construct an approximation of h^{**} from this estimate and then use it in their splitting algorithm. They apply their method to a tandem queue with two or three node and obtain good results. However, this method appears to have limited applicability for large and complicated models.

Booth and Hendricks [1984] propose adaptive methods that *learn* the importance function as follows. In their setting, the state space is partitioned in a finite number

of regions and the importance function h is assumed to be constant in each region. This importance function is used to determine the expected splitting factors and Russian roulette probabilities (see Section 2.8) when a chain jumps from one region to another. They estimate the "average" value of $\gamma(x)$ in each region by the fraction of chains that reach B among those that have entered this region. These estimates are taken as importance functions in further simulations used to improve the estimates, and so on.

Constructing the functions h^* or h^{**} essentially requires the knowledge of the probability $\gamma(x)$ for all x. But if we knew these probabilities, there would be no need for simulation! This is very similar (and related) to the issue of constructing the optimal change of measure in importance sampling [Glasserman et al. 1998]. In general, finding an optimal h, or an h for which $p_k(x)$ is independent of x, can be extremely difficult or even impossible. When $p_k(x)$ depends on x, selecting the thresholds so that $p_k \approx e^{-2}$ is not necessarily optimal. More importantly, with a bad choice of h, splitting may even *increase* the variance, as illustrated by the next example, taken from Glasserman et al. [1998] and Glasserman et al. [1999] and studied by several other authors.

For RESTART, Villén-Altamirano et al. [1994] concluded from a crude analysis that $p_k \approx e^{-2}$ was approximately optimal. However, their more careful analysis in Villén-Altamirano and Villén-Altamirano [2002] indicates that the p_k's should be as small as possible. Since the splitting factor at each level must be an integer, they recommend $p_k = 1/2$ and a splitting factor of $c_k = 2$.

Example 3. Consider an open tandem Jackson network with two queues, arrival rate 1, and service rate μ_j at queue j for $j = 1, 2$. Let $X_j = (X_{1,j}, X_{2,j})$ denote the number of customers at each of the two queues immediately after the jth event (arrival or end of service). We have $A = \{(0, 0)\}$ and $B = \{(x_1, x_2) : x_2 \geq \ell^*\}$ for some large integer ℓ^*. A naive choice of importance function here would be $h(x_1, x_2) = x_2$. This seems natural at first sight because the set B is defined in terms of x_2 only. With this choice, the entrance distribution at level k turns out to be concentrated on pairs (x_1, x_2) with small values of x_1. To see why, suppose that $x_2 = \ell_{k'} > 0$ for some integer k' and that we are in state $(x_1, x_2 - 1)$ where $x_1 > 0$ is small. The possible transitions are to states $(x_1 + 1, x_2 - 1)$, $(x_1, x_2 - 2)$, and $(x_1 - 1, x_2)$, with probabilities proportional to 1, μ_2, and μ_1, respectively. But the chains that go to state $(x_1 - 1, x_2)$ are cloned whereas the other ones are not, and this tends to increase the population of chains with a small x_1.

Suppose now that $\mu_1 < \mu_2$ (the first queue is the bottleneck). In this case, the most likely paths to overflow are those where the first queue builds up to a large level and then the second queue builds up from the transfer of customers from the first queue [Heidelberger 1995]. The importance function $h(x_1, x_2) = x_2$ does not favor these types of paths; it rather favors the paths where x_1 remains small and these paths have a very high likelihood of returning to $(0, 0)$ before overflow. As a result, splitting with this h may give an even larger variance than no splitting at all. Garvels et al. [2002] approximate h^{**} for this particular example; this h^{**} increases in both x_1 and x_2.

2.7 Unbiased Truncation

We pointed out earlier that a large fraction of the work in multilevel splitting is to simulate the chains down to level zero at each stage. Truncating the chains whenever they fall below some level $\ell_{k-\beta}$ in stage k, as explained earlier, reduces the work but introduces a bias. A large β may give negligible bias, but also a small work reduction. In what follows, we propose and examine unbiased truncation techniques, based on the *Russian roulette* principle [Kahn and Harris 1951; Hammersley and Handscomb 1964] and derandomized variants of it.

We use the following notation: Given two functions $f : \mathbb{R} \to [0, \infty)$ and $g : \mathbb{R} \to [0, \infty)$, we say that f is in $\mathcal{O}(g(x))$ [resp., in $\underline{\mathcal{O}}(g(x))$, in $\Theta(g(x))$] if $f(x) \le c_2 g(x)$ [resp., $c_1 g(x) \le f(x)$, $c_1 g(x) \le f(x) \le c_2 g(x)$] when $x \to \infty$, for some positive constants c_1 and c_2.

If A and B are fixed, m levels are placed optimally between them, and we use fixed-effort with n chains at each stage, then the work per stage in multilevel splitting is in $\Theta(n)$ and the total work is in $\Theta(nm)$. If $B = [\ell, \infty)$, then the work per stage and the total work also increase with ℓ, in a way that depends on the Markov chain model. The increase is not necessarily linear in ℓ, because the chain may move toward A much faster (on average) in some areas of the state space than in other areas. In any case, the total work is at least in $\underline{\mathcal{O}}(nm)$. The total work remains in $\underline{\mathcal{O}}(nm)$ with the proposed truncation methods, but with a smaller hidden constant.

Keeping a few representatives. A simple idea that may come to mind to remove the truncation bias is to keep a *small number of representatives* for all the chains that reach $\ell_{k-\beta}$ in stage k, for some integer $\beta \ge 2$. (Recall that in stage k, we start from ℓ_{k-1} and want to reach ℓ_k.) We can select a small integer $r \ge 1$. If M_k chains reach level $\ell_{k-\beta}$ downward in stage k, we select $r' = \min(M_k, r)$ of those chains at random, uniformly over the M_k chains. The selected chains are simulated until they reach either ℓ_k or 0, and the other ones are killed. Each of these r' selected chains *represents* M_k/r' chains, so it is given a *weight* of M_k/r'. If it reaches level ℓ_k, we clone it immediately in $\lfloor M_k/r' \rfloor + 1$ copies with probability δ and in $\lfloor M_k/r' \rfloor$ copies with probability $1 - \delta$, where $\delta = M_k/r' - \lfloor M_k/r' \rfloor$. These cloned chains are added to the chains that have reached level ℓ_k without going down to $\ell_{k-\beta}$, and R_k is the total number of all those chains, for $k = 1, \ldots, m$. The final estimator can defined exactly as before: $\hat{\gamma}_n = (R_1/N_0)(R_2/N_1) \cdots (R_m/N_{m-1})$. This estimator is unbiased, but it has a larger variance than the original one, because fewer chains are simulated from level $\ell_{k-\beta}$.

One practical difficulty in the implementation is to select the r' chains at random, because M_k is unknown until all the chains have reached either ℓ_k or $\ell_{k-\beta}$. One solution is to halt temporarily all the chains that hit $\ell_{k-\beta}$ until we know M_k. But this strategy is not compatible with the array-RQMC method discussed in Section 3. A solution that does not stop the chains can proceed as follows. Keep an index to the chains that reach level $\ell_{k-\beta}$, implemented as an array. Whenever a new chain reaches level $\ell_{k-\beta}$ for the first time in this stage, if there are already c chains in the index, placed at positions $0, \ldots, c-1$, we generate a random integer I in $\{0, \ldots, c\}$. The chain in position I is moved to position c and the new chain is placed at position I. There are now $c+1$ chains in the index. In the end, we keep only the r'

chains that are in positions $\{0, \ldots, r'-1\}$; they are the first r' elements of a random permutation of the M_k chains. The chains that are in positions $\{r, r+1, \ldots\}$ of the index can in fact be killed at any time, and their contributions must be removed if they have already hit ℓ_k, but this killing should not affect the counter c. There is no need to know the value of M_k beforehand, i.e., we do not have to wait until all the M_k chains have reached level $\ell_{k-\beta}$ before selecting the representatives. On the other hand, a lot of work is often wasted with this second approach, because chains can be simulated for several steps after they hit $\ell_{k-\beta}$ before being killed. If r is large, not much work is wasted but not much work is saved either. If r is small (e.g., $r = 1$ in the extreme case), then each representative has a large weight and this increases the variance; it becomes a rare event that a representative reaches ℓ_k. Thus, even though the bias has been removed in theory, the contribution of the representative chains to the estimator has itself a large relative error. If we replicate the splitting scheme a few times with a large enough β, it is likely that none of the representatives will contribute, in which case the estimator is exactly the same as with (biased) truncation alone.

To reduce the variance contribution of the representatives, we could define a hierarchical version of this method: At each level ℓ_{k-j}, for $j \geq \beta$, we select a maximum of $r_{k,j}$ representatives at random among the chains that have reached this level, where $1 \leq r_{k,k-1} \leq r_{k,k-2} \leq \cdots \leq r_{k,\beta}$. With good choices of β and the $r_{k,j}$, and careful management of the weights, this method provides an unbiased estimator, conceivably with smaller variance than the previous one, because there is more variety in the trajectories. A major drawback, however, is that its implementation requires very complicated bookkeeping, and combining it with array-RQMC is practically a non-starter.

We now introduce variants of this idea that are much easier to implement. In all these variants, we use $\tilde{R}_k$ to denote the sum of weights of all the chains that reach level ℓ_k, *before* the cloning (when they reach that level). These $\tilde{R}_k$ are not necessarily integers. There are two types of cloning at the end of a stage k: (1) a chain with weight w is cloned into a random number of copies with expectation w (type-1 cloning) and (2) the chains are then cloned again (if needed) so that their total number reaches the target value N_k (type-2 cloning). We use R_k to denote the total number of chains after the cloning of type 1, but before the cloning of type 2. Thus, each R_k is a random integer, $\mathbb{E}[R_k \mid \tilde{R}_k] = \tilde{R}_k$, and $\mathrm{Var}[R_k] \geq \mathrm{Var}[\tilde{R}_k]$. The final estimator can be defined either as before:

$$\hat{\gamma}_n = (R_1/N_0)(R_2/N_1)\cdots(R_m/N_{m-1}),$$

or by

$$\hat{\gamma}_n = (\tilde{R}_1/N_0)(\tilde{R}_2/N_1)\cdots(\tilde{R}_m/N_{m-1}).$$

This second estimator is the conditional expectation of the first one, conditional on $(\tilde{R}_1, \ldots, \tilde{R}_m)$, so it necessarily has smaller (or equal) variance. For this reason, this is the one we recommend and use.

Probabilistic truncation. The following simplified version of the preceding method requires much less bookkeeping. Instead of keeping representatives, just kill the chains at random, with some probability, independently of each other. For stage k, we select real numbers $r_{k,\beta}, \ldots, r_{k,k-1}$ in $[1, \infty)$. The first time a chain reaches

level ℓ_{k-j} from above during that stage, for $j \geq \beta$, it is killed with probability $1 - 1/r_{k,j}$. If it survives, its weight is multiplied by $r_{k,j}$. (This is a version of Russian roulette.) When a chain of weight $w > 1$ reaches level ℓ_k, it is cloned into $w - 1$ additional copies and each copy is given weight 1 (if w is not an integer, we make $\lfloor w \rfloor$ additional copies with probability $\delta = w - \lfloor w \rfloor$ and $\lfloor w - 1 \rfloor$ additional copies with probability $1 - \delta$). Now, the number of representatives kept at any given stage is random; it has a binomial distribution. Without loss of generality, we can now assume that $\beta = 2$, because selecting a larger β is equivalent to taking $r_{k,j} = 1$ for $j \leq \beta$. We will do that for the remainder of the paper.

Periodic truncation. To reduce the variability of the number of selected representatives at each level ℓ_{k-j}, we may decide to retain every $r_{k,j}$-th chain that down-crosses that level; e.g., if $r_{k,j} = 3$, we keep the third, sixth, ninth, etc. This would generally give a biased estimator, because the probability that a chain is killed would then depend on its sample path up to the time when it crosses the level (for instance, the first chain that down-crosses the level would always be killed if $r_{k,j} > 1$). A simple trick to remove that bias is to modify the method as follows: generate a random integer $D_{k,j}$ uniformly in $\{1, \ldots, r_{k,j}\}$, retain the $(i\, r_{k,j} + D_{k,j})$-th chain that down-crosses level ℓ_{k-j} for $i = 0, 1, 2, \ldots$, and kill the other ones. We assume that the random variables $D_{k,2}, \ldots, D_{k,k-1}$ are independent. Then, any chain that down-crosses the level has the same probability $1 - 1/r_{k,j}$ of being killed, independently of its trajectory above that level. This is true for any positive integer $r_{k,j}$. Moreover, the proportion of chains that survive has less variance than for the probabilistic truncation (the killing indicators are no longer independent across the chains). The chains that reach ℓ_k are cloned in proportion to their weight, exactly as in the probabilistic truncation.

Tag-based truncation. In the periodic truncation method, the level at which a chain is killed is determined only when the chain reaches that level. An alternative is to fix all these levels right at the beginning of the stage. We first select positive integers $r_{k,2}, \ldots, r_{k,k-1}$. Then each chain is *tagged* to the level ℓ_{k-j} with probability $q_{k,j} = (r_{k,j} - 1)/(r_{k,2} \cdots r_{k,j})$ for $j = 2, \ldots, k-1$, and to level ℓ_0 with probability $1 - q_{k,k-1} - \cdots - q_{k,2} = 1/(r_{k,2} \cdots r_{k,k-1})$. Thus, all the chains have the same probability of receiving any given level and the probability of receiving level zero is positive. If the tags are assigned randomly and independently across the chains, then this method is equivalent to probabilistic truncation. But if the integers $r_{k,2}, \ldots, r_{k,k-1}$ are chosen so that their product divides (or equals) N_k, the number of chains at the beginning of stage k, then the tags can also be assigned so that the proportion of chains tagged to level ℓ_{k-j} is *exactly* $q_{k,j}$, while the probability of receiving a given tag is the same for all chains. The reader can verify that the following scheme gives one way of achieving this: Put the N_k chains in a list (in any order), generate a random integer D uniformly in $\{0, \ldots, N_k - 1\}$, and assign the tag $k - j^*(i, D)$ to the i-th chain in the list, for all i, where $j^*(i, D)$ is the smallest integer j in $\{2, \ldots, k\}$ such that $r_{k,2} \cdots r_{k,j}$ *does not* divide $(D + i)$ mod N_k (when $(D + i)$ mod $N_k = 0$, we put $j^*(i, D) = k$). After the tags are assigned, the chains can be simulated one by one for that stage. Whenever a chain down-crosses for the first time (in this stage) a level ℓ_{k-j} higher than its tag, its weight is multiplied by

$r_{k,j}$. If it down-crosses the level of its tag, it is killed immediately. The chains that reach ℓ_k are cloned in proportion to their weight, as before.

Unbiasedness. To prove that the above truncation methods are all unbiased, we will argue that each of them satisfies the assumptions of the following proposition.

PROPOSITION 2.1. *Suppose there are real numbers $r_{k,2}, \ldots, r_{k,k-1}$ in $[1, \infty)$ such that for $j = 2, \ldots, k-1$, each chain has a probability $1 - 1/r_{k,j}$ of being killed at its first down-crossing of level ℓ_{k-j}, independently of its sample path up to that moment, and its weight is multiplied by $r_{k,j}$ if it survives. Then the truncated estimator remains unbiased.*

PROOF. We use a similar argument as in Booth and Pederson [1992, Section V]. We already know that the multilevel splitting estimator is unbiased, so it suffices to show that the truncation does not change the expectation. In fact, ordinary multilevel splitting is a special case of the scheme described in the proposition statement, with $r_{k,j} = 1$ for all j, at all levels. So if we can show that the expectation of the estimator with truncation does not depend on the $r_{k,j}$'s, the result will follow. We will do that by concentrating on a given stage k. The same argument applies to any of the stages, so we will be able to conclude that we can change the $r_{k,j}$'s at any of the stages without introducing a bias.

Consider a chain that down-crosses ℓ_{k-j} for the first time in stage k, for $j \geq 2$, in the setting of the proposition statement. Let $\tilde{E}_{k,j}$ be the expected contribution of that chain to the final estimator $\hat{\gamma}_n = \hat{p}_1 \cdots \hat{p}_m$ at this down-crossing moment, and let $\tilde{E}_k$ be its expected contribution at the time when it hits ℓ_k if it does (otherwise, $\tilde{E}_k = 0$). If the chain has weight W when it hits ℓ_k, then its expected contribution at that epoch is $\tilde{E}_k = WE_k$ where E_k is the expected contribution of each of its copies after the cloning. Then, since the weight of a chain has no influence on its sample path until it reaches ℓ_k, the expected contribution of a chain of weight W is W times the expected contribution of a chain of weight 1 in the same state. Note that $\tilde{E}_{k,j}$, $\tilde{E}_k$, and E_k are random variables (they are conditional expectations).

When a chain of weight W down-crosses ℓ_{k-j} for the first time, its expected contribution is the probability that it survives, times its expected contribution if it does. If we denote by $E_{k,j}$ the realization of $\tilde{E}_{k,j}$ when $W = r_{k,j} = 1$, recalling that the weight is multiplied by $r_{k,j}$ if the chain survives, we obtain

$$\tilde{E}_{k,j} = (1/r_{k,j})(r_{k,j}W)E_{k,j} = WE_{k,j},$$

which does not depend on $r_{k,j}$. Since this holds for all $j \geq 2$, the expected contribution depends on none of the $r_{k,j}$'s. (Note that we necessarily have $W = 1$ for $j = 2$.) This completes the proof. □

It remains to verify that the three proposed truncation methods satisfy the conditions of the proposition. For the probabilistic truncation, this is clear from its definition. For the periodic and tag-based truncation, if we look at a single chain when it down-crosses level ℓ_{k-j} for the first time and condition on its sample path up to that time (and no other information), the conditional probability that this chains survives this down-crossing is $1/r_{k,j}$, so the conditions are satisfied.

Getting rid of the weights. With the unbiased truncation methods discussed so far, the surviving chains have different weights. The variance of these weights may contribute significantly to the variance of the final estimator. For example, if k is large, the event that a chain reaches ℓ_k (from ℓ_{k-1}) after going down to ℓ_1 is usually a rare event, and when it occurs the corresponding chain has a large weight, so this may have a non-negligible impact on the variance. This can be addressed by resplitting the chains within the stage when they up-cross some levels, instead of increasing their weights at down-crossings. We now explain how the probabilistic and tag-based truncation methods can be modified to incorporate this idea. In these methods, the weights of all chains are always 1, and whenever a chain down-crosses ℓ_{k-j} (not only the first time), for $j \geq 2$, it can get killed.

Probabilistic truncation and resplitting within each stage. The probabilistic truncation method can be modified as follows. During stage k, whenever a chain reaches a level ℓ_{k-j} from below, it is split in $r_{k,j}$ identical copies that start evolving independently from that point onward (if $r_{k,j}$ is not an integer, we split the chain in $\lfloor r_{k,j} + 1 \rfloor$ copies with probability $\delta = r_{k,j} - \lfloor r_{k,j} \rfloor$ and in $\lfloor r_{k,j} \rfloor$ copies with probability $1 - \delta$). Whenever a chain down-crosses ℓ_{k-j} (not only the first time), for $j \geq 2$, it is killed with probability $1 - 1/r_{k,j}$. All chains always have weight 1.

Tag-based truncation with resplits. This method is equivalent to applying RESTART separately within each stage of the multistage splitting algorithm. It modifies the tag-based truncation as follows: Whenever a chain up-crosses level ℓ_{k-j} for $j \geq 2$, it is split in $r_{k,j}$ copies. One of these $r_{k,j}$ copies is identified as the original and keeps its current tag, while the other $r_{k,j} - 1$ copies are tagged to the level ℓ_{k-j} where the split occurs.

Unbiasedness. We will show that the proposed truncation methods with resplit are covered by the following proposition.

PROPOSITION 2.2. *Suppose there are positive real numbers $r_{k,2}, \ldots, r_{k,k-1}$ such that for $j = 2, \ldots, k-1$, each chain is killed with probability $1 - 1/r_{k,j}$ whenever it down-crosses level ℓ_{k-j}, independently of its sample path up to the time when it reached that level, and that this chain is split into C chains when it up-crosses that same level, where C is a random variable with mean $r_{k,j}$, independent of the history so far. Then the estimator with probabilistic truncation and resplits (without weights) is unbiased for γ_0.*

PROOF. We extend the proof of Proposition 2.1 to cover resplits. Again, it suffices to show that none of the expected contributions depends on the values of $r_{k,j}$ used in the future steps, in stage k.

Consider a chain that down-crosses ℓ_{k-j} in stage k. Its expected contribution to the final estimator, at that time, is $\tilde{E}_{k-j} = (1/r_{k,j})E_{k-j}$, where E_{k-j} is its expected contribution if it survives (a random variable). Let $E^{(c)}_{k-j}$ be its expected contribution E_{k-j} if we assume that there is no resplit at that level (i.e., if $r_{k,j} = 1$). Then, $E_{k-j} = r_{k,j}E^{(c)}_{k-j}$, and therefore $\tilde{E}_{k-j} = E^{(c)}_{k-j}$, which is independent of $r_{k,j}$. □

The probabilistic truncation with resplits obviously satisfies the assumptions of Proposition 2.2. The tag-based truncation with resplits does not seem to satisfy these assumptions at first sight, but it does if we see things from the right perspective. The trick is to hide the tags. Indeed, when a chain down-crosses a level ℓ_{k-j} for the first time, the probability that it is tagged to that level and gets killed is $1 - 1/r_{k,j}$. Otherwise, if it is not the first time, then this chain is one of the $r_{k,j}$ copies made when a chain up-crossed that level earlier. This chain will survive the down-crossing if and only if it is *not* tagged to level ℓ_{k-j}, if and only if it was the original when these copies were made, and this happens with probability $1/r_{k,j}$.

We could also consider *periodic truncation with resplits*, but this method does not satisfy the assumptions of Proposition 2.2 and we have no proof that it is unbiased.

Comparison, implementation, and choice of the $r_{k,j}$'s. The resplit versions of the methods are expected to give a smaller variance but require more work. So there is no universal winner if we think of maximizing the efficiency. Another disadvantage of this resplit strategy is that the *number* of chains alive at any given time during stage k will now have more variance and *may* exceed N_{k-1}. In a *worst-case* situation, a chain may go down and up many times across several levels without being killed, giving rise to a flurry of siblings along the way. Fortunately, this type of bad behavior has an extremely small probability and poses no problem when the splitting parameters are well chosen. In all the experiments that we have run, the number of chains alive simultaneously during any given stage k has rarely exceeded N_{k-1}. In our implementation, whenever the number of chains was going to exceed N_{k-1}, we used weights instead of splitting, just for the splits that would have made the number of chains too large. We did that because our chains were stored in an array of size $n = N_{k-1}$ and we did not want their number to exceed n. This type of implementation is needed when we combine the splitting with array-RQMC (see Section 4).

We have a lot of freedom for the choice of the truncation and resplit parameters $r_{k,j}$. We can also select different sets of values at the different stages of the multilevel splitting algorithm. Empirically, we found that starting the truncation too soon was counterproductive; it was better to take $r_{j,k} = 1$ for $j \le 2$. For $j \ge 3$, it appears sensible to take $r_{k,j} = 1/\hat{p}_{k-j} = N_{k-j-1}/R_{k-j}$, the actual splitting factor used at level ℓ_{k-j} of the splitting algorithm. Another possibility is $r_{k,j} = \hat{\gamma}_{(k-1)}^{-1/(k-1)}$, where $\hat{\gamma}_{(k-1)} = \hat{p}_1\hat{p}_2\dots\hat{p}_{k-1}$. We tried these two choices in our experiments and they both worked well.

2.8 Getting Rid of the Levels

In some versions of the *splitting and Russian roulette* technique, there are no levels (or thresholds), but only an importance function (some authors call it *branching function*). For instance, Ermakov and Melas [1995] and Melas [1997] study a general setting where a chain can be split or killed at any transition. If the transition is from x to y and if $\alpha = h(y)/h(x) \ge 1$, then the chain is split in C copies where $\mathbb{E}[C] = \alpha$, whereas if $\alpha < 1$ it is killed with probability $1 - \alpha$ (this is Russian roulette). In case of a split, the $C - 1$ new copies are started from state x and new transitions are generated (independently) for those chains. Their method is developed to estimate the average cost per unit of time in a regenerative process,

where a state-dependent cost is incurred at each step. In the simulation, each cost incurred in a given state x is divided by $h(x)$. We may view $1/h(x)$ as the *weight* of the chain at that point. At the end of a regenerative cycle, the total weighted cost accumulated by the chain over its cycle is the *observation* associated with this cycle. The expected cost per cycle is estimated by averaging the observations over all simulated cycles. The expected length of a cycle is estimated in the same way, just replacing costs by lengths. The authors show that their method is consistent and propose an adaptive algorithm that estimates the optimal h.

This method can be applied to a finite-horizon simulation as well. In our setting, it suffices to replace the regeneration time by the time when the chain reaches A or B, and to forget about the length of the cycle. When a chain reaches B, it contributes its weight $1/h(X_{\tau_B})$ to the estimator. For a very crude analysis, suppose we take $h(x) = \gamma(x)$ and that there is a split in two every time the function h doubles its value. Here, $h(y)/h(x) = \gamma(y)/\gamma(x)$, so a chain that reaches the set B would have split in two approximately $-\log_2 \gamma_0$ times. This gives a "potential" of $2^{-\log_2 \gamma_0} = 1/\gamma_0$ copies that can possibly reach B for each initial chain at level 0, the same number as for the multilevel splitting and RESTART; see Equation (6). This argument suggests that an optimal h in this case should be approximately proportional to $\gamma(x)$.

In general, splitting and Russian roulette can be implemented by maintaining a weight for each chain. Initially, each chain has weight 1. Whenever a chain of weight w is split in C copies, the weight of all the copies is set to either w/C or $w/\mathbb{E}[C]$. Booth [1985] shows that using $w/\mathbb{E}[C]$ is usually better. When Russian roulette is applied, the chain is killed with some probability $\alpha < 1$; if it survives, its weight is multiplied by $1/(1-\alpha)$. The values of C and α at each step can be deterministic or random, and may depend on the past history of the chain. Whenever a cost is incurred, it must be multiplied by the weight of the chain. Unbiasedness for this general setting is proved (under mild conditions) by Booth and Pederson [1992], for example.

2.9 Weight Windows

Particle transport simulations in nuclear physics often combine splitting and Russian roulette with importance sampling. Then, the weight of each chain must be multiplied by the *likelihood ratio* accumulated so far. The *weight* is redefined as this product. In the context of rare events, it is frequently the case that the final weight of a chain is occasionally large and usually very small. This gives rise to a large variance and a highly-skewed distribution, for which variance estimation is difficult.

To reduce the variance of the weights, Booth [1982] introduced the idea of *weight windows*, which we define as follows (see also Booth and Hendricks [1984] and Fox [1999]). Define the *weighted importance* of a chain as the product of its weight w and the value of the importance function $h(x)$ at its current state. Select three real numbers $0 < a_{\min} < a < a_{\max}$. Whenever the weighted importance $\omega = w\,h(x)$ of a chain falls below $a_{\min}$, we apply Russian roulette, killing the chain with probability $1-\omega/a$. If the chain survives, its weight is set to $a/h(x)$. If the weighted importance ω rises above $a_{\max}$, we split the chain in $c = \lceil \omega/a_{\max} \rceil$ copies and give weight w/c to each copy. The estimator of $\gamma_0 = \mathbb{P}[\tau_B < \tau_A]$ is the sum of weights of all the chains that reach the set B before reaching A. The importance function $h^*(x) = \gamma(x)$

should be approximately optimal in this case. The basic motivation is simple: if the weight window is reasonably narrow, all the chains that reach B would have approximately the same weight, so the only significant source of variance would be the *number* of chains that reach B [Booth and Hendricks 1984]. If we take $a = (a_{\min} + a_{\max})/2 \approx \mu$, then this number has expectation n (approximately), where n is the initial number of chains.

In the original proposal of Booth [1982] and Booth and Hendricks [1984], the windows are on the weights, not on the weighted importance. The state space is partitioned in a finite number of regions (say, up to 100 regions), the importance function is assumed constant in each region, and each region has a different weight window, inversely proportional to the value of the importance function in that region. Such weight windows are used extensively in the Los Alamos particle transport simulation programs. Our formulation is essentially equivalent, except that we do not assume a finite partition of the state space.

Fox [1999, Chapter 10] discusses the use of weight windows for splitting and Russian roulette, but does not mention the use of an importance function. Weight windows without an importance function could be fine when a good change of measure (importance sampling) is already applied to drive the system toward the set B. Then, the role of splitting and Russian roulette is only to "equalize" the contributions of the chains that reach B and kill most of those whose anticipated contribution is deemed negligible, to save work. This type of splitting, based only on weights and without an importance function, gives no special encouragement to the chains that go toward B. If we use it alone, the event $\{\tau_B < \tau_A\}$ will remain a rare event.

If there is no importance sampling, the multilevel splitting techniques described earlier (except those with truncation and no resplits, in Section 2.7) have the advantage of not requiring explicit (random) weights. All the chains that reach level ℓ_k have the same weight when they reach that level for the first time. So there is no need for weight windows in that context.

3. RQMC AND ARRAY-RQMC FOR SIMULATING MARKOV CHAINS

Monte Carlo for a Stochastic Recurrence. It is convenient to write our Markov chain $\{X_j,\ j \geq 0\}$ as a stochastic recurrence of the form

$$X_{j+1} = \varphi_j(X_j, \mathbf{U}_j) \tag{7}$$

where the $\mathbf{U}_j$'s are independent random vectors uniformly distributed over $[0,1)^d$ and $X_0 = x_0$. If random numbers are required for splitting and Russian roulette decisions in step j, these random numbers are assumed to be part of $\mathbf{U}_j$. When simulating this Markov chain, these $\mathbf{U}_j$ are "realized" by a random number generator. To simulate the chain until it reaches A or B, we need a vector of τd independent random numbers uniformly distributed over $[0,1)$.

Let s (an integer) be an upper bound on τd, over all possible sample paths. If no such upper bound exists, take $s = \infty$. The random variable $Y = \mathbb{I}[\tau_B < \tau_A]$, where $\mathbb{I}$ denotes the indicator function, can be written as $Y = f(U_1, \ldots, U_s)$ for some function f, where the U_l's are independent $U(0,1)$ random variables and $\mathbf{U}_j = (U_{(j-1)d+1}, \ldots, U_{jd})$ for each j. The standard *Monte Carlo (MC) method* generates n independent random points $\mathbf{V}_i = (U_{i,1}, \ldots, U_{i,s})$ uniformly distributed

in the s-dimensional unit cube $[0,1)^s$, for $i = 1, \ldots, n$, computes $Y_i = f(\mathbf{V}_i)$ for each i, and estimates μ by the average $\bar{Y}_n$ of these n numbers. (In practice, only the coordinates of $\mathbf{V}_i$ that are needed are actually generated.)

Classical RQMC. The aim of the *classical RQMC* method [Owen 1998; L'Ecuyer and Lemieux 2002, and references therein] is to induce "negative dependence" between the $\mathbf{V}_i$'s to reduce the variance of the average $\bar{Y}_n$. This is the same general idea as *generalized antithetic variates* [Wilson 1983]. The (random) set $P_n = \{\mathbf{V}_1, \ldots, \mathbf{V}_n\}$ is called an *RQMC point set* if (a) each $\mathbf{V}_i$ has the uniform distribution over $[0,1)^s$ and (b) P_n covers $[0,1)^s$ "more uniformly" than a set of independent random points, with probability 1. The precise meaning of "more uniformly" is left open voluntarily; to complete the definition, we must adopt a specific measure of uniformity and a wide variety of such measures are available [Niederreiter 1992; L'Ecuyer and Lemieux 2002]. Examples of RQMC point sets include randomly shifted lattice rules, scrambled digital nets, digital nets with a random digital shift, Latin hypercube samples, etc. [Owen 1998; L'Ecuyer and Lemieux 2002]. Now that the Y_i's are dependent, we cannot estimate their variance by their sample variance. But we can replicate this entire scheme R times, independently (e.g., with independent randomizations of the same point set) and estimate the variance of $\bar{Y}_n$ by the sample variance of its R copies. This RQMC approach can be quite efficient when s is small or when f has low effective dimension in some sense [Owen 1998; L'Ecuyer and Lemieux 2002]. In some contexts, however, the effective dimension is large. In our setting, there is usually no finite upper bound on the stopping time τ, so we must take $s = \infty$.

Array-RQMC. L'Ecuyer et al. [2006] recently proposed a different RQMC method for Markov chains, based on an earlier (deterministic) QMC algorithm of Lécot and Tuffin [2004]. They call it *array-RQMC*. It simulates n copies of the chain in parallel, using a randomized highly-uniform point set to generate the next state of these n chains at each step, and reorders the chains after each step. This reordering is made according to the value of an *importance function* $v : \mathcal{X} \to \mathbb{R}$ at the current state of each chain. When splitting is combined with array-RQMC, this function v is not necessarily the same as the importance function h used for the splitting; so here we call v the *sorting function*, to avoid confusion.

Thus, the method operates as follows. At step 1, we select an RQMC point set $P_{n,1} = \{\mathbf{u}_{0,1}, \ldots, \mathbf{u}_{n-1,1}\}$ in $[0,1)^d$, and define

$$X_{i,1} = \varphi_1(x_0, \mathbf{u}_{i,1}) \qquad \text{for } i = 0, \ldots, n-1.$$

After this step, we expect that the empirical distribution of $X_{0,1}, \ldots, X_{n-1,1}$ is a better approximation of the distribution F_1 of X_1 than with independent random points instead of $P_{n,1}$. In case where the initial state of the chain should be generated from some initial distribution instead of being fixed, we can simply view $X_{i,1}$ as the random initial state.

At *step* j, we take an RQMC point set $P_{n,j} = \{\mathbf{u}_{0,j}, \ldots, \mathbf{u}_{n-1,j}\}$ in $[0,1)^d$ constructed so that the $(d+1)$-dimensional *modified RQMC point set* $P'_{n,j} = \{\mathbf{u}'_{i,j} = ((i + 0.5)/n, \mathbf{u}_{i,j}),\ 0 \le i < n\}$ is "highly uniform" in $[0,1)^{d+1}$, in a sense that we leave open (as in the definition of RQMC point set). This $P'_{n,j}$ can be defined by

taking a $(d+1)$-dimensional RQMC point set, sorting the points by order of their first coordinate, and replacing the first coordinate of the ith point by $(i+0.5)/n$ for each i. Usually, $P_{n,j}$ is obtained by re-randomizing *the same* point set at all steps j. Let $X_{(0),j-1},\ldots,X_{(n-1),j-1}$ be the states of the chains immediately before step j, sorted by increasing order of $v(X_{i,j-1})$. When a chain reaches its stopping time, we put it in a special state x for which $v(x)=\infty$. For $i=0,\ldots,n-1$, we define $X_{i,j}=\varphi_j(X_{(i),j-1},\mathbf{u}_{i,j})$ if $v(X_{(i),j-1})<\infty$ and $X_{i,j}=X_{(i),j-1}$ otherwise. Then these states are sorted by increasing value of $v(X_{i,j})$ and we go to step $j+1$. The intuitive motivation of the method is that we expect the empirical distribution of $X_{(0),j},\ldots,X_{(n-1),j}$ to better approximate the theoretical distribution F_j of X_j than if we were simulating everything with independent uniform random numbers. So, if an estimator is the average of state-dependent costs, averaged over the steps j and over the n chains, we expect (heuristically) the variance to be smaller with array-RQMC than with independent random numbers.

L'Ecuyer et al. [2005] prove that the array-RQMC estimator is unbiased, and provide a *worst-case* bound of $\mathcal{O}(n^{-1/2})$ for the variance, in the case of a one-dimensional state space and under some conditions. For a special case of the method that corresponds essentially to stratification, again with a one-dimensional state space, they show that the variance converges to 0 as $\mathcal{O}(n^{-3/2})$. An unbiased variance estimate can be obtained by replicating the entire scheme R times independently, as for classical RQMC. For additional justifications and details, and illustrations of the degrees of improvement that are obtained in practice, we refer the reader to L'Ecuyer et al. [2005].

4. COMBINING SPLITTING WITH RQMC AND ARRAY-RQMC

Our primary interest in this paper is the combination of splitting or IS with RQMC or array-RQMC, with a special attention to the splitting/array-RQMC combination. We now explain how multilevel splitting can be combined first with classical RQMC, then with array-RQMC.

Classical RQMC for fixed-effort multilevel splitting, without resplits within a stage. We can apply the classical RQMC method at each stage of the multilevel splitting, as follows, assuming that there are no resplits during the stage. At the beginning of stage k, we select an infinite-dimensional RQMC point set $P_{N_{k-1}}$ of cardinality N_{k-1} (in the sense given in Section 3), and randomize it. Each of the N_{k-1} chains is simulated by using the successive coordinates of one point of $P_{N_{k-1}}$, until it reaches ℓ_k or ℓ_0 or gets killed. This is usually implemented by simulating one chain at a time, doing the entire trajectory of the chain before going to the next one. If there is no truncation during that stage, these successive coordinates are used to determine the successive transitions of the chain as explained in Section 3. The RQMC point set is taken as infinite-dimensional because there is usually no upper bound on the number of steps before reaching ℓ_k or ℓ_0. This limits the range of choices for this point set; for example, Sobol' point sets, a particular class of digital nets, are ruled out. If we use truncation and weights within the stage, as in Section 2.7, then we also need random numbers to determine the truncation decisions. These additional random numbers can be taken either (a) from the successive coordinates of the RQMC point as well, or (b) from an independent RQMC

point set, or (c) from an independent stream of random numbers. None of these approaches universally dominates the others; it depends on the model. In all cases, the successive random numbers used for any given chain are independent uniform random variables over $[0, 1)$, so the expected contribution of each chain to the estimator is the same as with MC, which shows that the method is unbiased. For the experiments reported in Section 5, we have used (c).

Combining array-RQMC with splitting. The basic idea here is to take one of the several variants of splitting discussed in Section 2, implement it in a *breath-first* fashion, and simulate all the chains in parallel, one step at a time, using array-RQMC. For multilevel splitting, the array-RQMC is applied separately for each stage to estimate the corresponding probability p_k. Whenever a chain reaches the next level, it is put in a "special state" and waits there until the stage is over. At the end of the stage, we split the chains that have reached the target level ℓ_k and restart the array-RQMC algorithm from there.

Hurdles. A first difficulty appears immediately: with splitting, the number of chains alive at any given step is random, so it cannot be taken always equal to a given constant n as in the plain vanilla array-RQMC algorithm. Moreover, for several splitting variants (such as multilevel fixed-splitting,

RESTART, and splitting without levels as in Section 2.8), if we start with n chains, the number of simultaneous chains at a later step could grow eventually much larger than n. What do we do with array-RQMC in this case? In principle, we could use an RQMC point set P_n with different cardinality n at each step, equal to the number of chains alive at that step. But creating those different RQMC point sets would introduce significant overhead (as opposed to creating a single one of cardinality n and reusing it with an independent randomization at each step), because the creation usually involves a one-time setup which is much more computationally expensive than enumerating the points. Multilevel splitting with fixed effort appears to be the variant best adapted to array-RQMC, because (at least in its basic version) the number of chains never exceeds the selected constant N_k at each stage k. This constant can be taken always equal to n. If we combine the fixed-effort multilevel splitting with any of the "resplit-type" unbiased truncation methods discussed in Section 2.7, the difficulty of an unbounded number of chains pops up again because of the resplitting within each stage. One possibility is to temporarily cut off the splitting when there are n chains alive and increase the weight of chains instead of splitting them, as discussed earlier. When the number of chains becomes less than n, splitting is resumed (the number of copies to make in a resplit is multiplied by the weight of a chain).

Array-RQMC for fixed-effort multilevel splitting, without truncation. This basic case is the one for which array-RQMC is easiest to implement. The combined method operates as follows. Let $X^{(k)}$ denote the Markov chain $\{X_j,\, j \geq 0\}$ between times T_{k-1} and $\min(\tau_A, T_k)$. We start stage 1 with N_0 chains. We estimate p_1 by using the array-RQMC algorithm for the Markov chain $X^{(1)}$: The N_0 chains are simulated in parallel according to (7) and sorted after each time step. Each chain evolves until the stopping time $\min(T_1, \tau_A)$. If R_1 is the number of chains for which D_1 occurs, R_1/N_0 is an unbiased estimator of p_1. The states of these R_1

chains are stored and their empirical distribution can be viewed an estimator of the distribution G_1 of X_{T_1}.

At the second stage, N_1 chains are started from those R_1 states according to the fixed-effort splitting policy, and are simulated in parallel using the same array-RQMC procedure on $X^{(2)}$, each chain being simulated until its stopping time $\min(T_2, \tau_A)$. The probability p_2 is estimated by R_2/N_1 where R_2 is the number of chains for which D_2 occurs. These R_2 chains are then split again, and so on, until all the probabilities $p_3, \cdots, p_m$ have been estimated.

The algorithm is given in Figure 1. It basically consists in adding a loop to the algorithm of L'Ecuyer et al. [2005] for the estimation of the probability p_k of reaching each successive level. In this algorithm, $X_{i,j}^{(k)}$ represents the state of the ith chain at step j, in stage k. For simplicity, we consider a single replication. This entire procedure must be repeated R times with independent randomizations to estimate the variance and compute a confidence interval on γ_0. As in L'Ecuyer et al. [2006], it can be readily verified that the estimator of each p_k is unbiased, and we have:

PROPOSITION 4.1. *The estimator $\tilde{\gamma}_n$ in Figure 1 has expectation γ_0.*

PROOF. The randomization of the point set before each step ensures that the sample path of any chain, considered separately, is determined by a sequence of independent random numbers uniformly distributed in $[0, 1)$. Therefore, each chain taken individually has exactly the same probability law as with splitting with MC, and has the same expected contribution to the estimator. Combining this with (3), which says that the splitting estimator is unbiased, we get the result. □

Unbiased truncation and potential difficulties. The truncation methods of Section 2.7, without the resplits, are easy to incorporate to the algorithm of Figure 1. Each chain now has a weight and when a chain is killed it is put into a state x with $v(x) = \infty$. The number of chains considered by the inner loop would then decrease with j faster than without truncation. In the case of resplits, we must make sure that the number of chains alive never exceeds N_{k-1} in stage k. A solution to this problem was given previously. There are other possibilities, such as enlarging the RQMC point set when needed, using independent random numbers for the extra chains, etc.

The algorithm of Figure 1 is rather simple in principle. However, several practical issues must be addressed for the combination of splitting and array-RQMC to be really effective. Some of these difficulties are discussed in the following remarks.

(1) Recall that in a given stage, the number of time steps before reaching the next level or coming back to A is random, can be large, and may have large variability, especially when k is large, because the chain then starts far from A and requires a large number of steps to come back to A. For the array-RQMC method, this causes the number of points used in successive steps to decrease with the step number during that stage, thus reducing the RQMC efficiency because only a few points from the RQMC point set are used in the later steps.

(2) In the case of random resplits and roulette, we need additional random numbers. If we use separate independent random numbers for that, this would add a vari-

Initialization.
For $k = 1, \ldots, m$, select a d-dimensional RQMC point set $P_{N_{k-1}}^{(k)}$ constructed so that the corresponding $P_{N_{k-1}}^{\prime(k)}$ is a modified RQMC point set;
Select the m thresholds $0 = \ell_0 < \ell_1 < \cdots < \ell_m = \ell$;
Initialize the N_0 chains according to the initial state distribution;

Estimate each p_k.
For $(k = 1;\ k <= m;\ k{+}{+})$
    Simulate in parallel N_{k-1} copies of the chain, as follows:
    If $k > 1$, split the R_{k-1} available chains to get N_{k-1} chains
        according to the splitting policy;
    For $(j = 1;\ v(X_{0,j-1}^{(k)}) < \infty;\ j{+}{+})$
        Sort (and renumber) the chains by increasing order of $v(X_{i,j-1}^{(k)})$;
        Randomize $P_{k,N_{k-1}}$ afresh;
        For $(i = 0;\ i < N_{k-1}$ and $X_{i,j-1}^{(k)} < \infty;\ i{+}{+})$
            $X_{i,j}^{(k)} = \varphi_j(X_{i,j-1}^{(k)}, \mathbf{u}_i)$;

Output.
Return $\tilde{\gamma}_n = \prod_{k=1}^{m} R_k/N_{k-1}$ as an estimator of γ_0.

Fig. 1. Fixed-effort multilevel splitting (basic version) combined with array-RQMC

ance component and may dilute the gain obtained by array-RQMC. Another possibility is to add dimensions to the RQMC point set and use the additional coordinates for the splitting and roulette decisions. But in general, increasing the dimension tends to reduce the effectiveness of RQMC. This favors the methods where the splitting and roulette decisions are more deterministic, i.e., the tag-based and periodic approaches, as opposed to probabilistic truncation and resplit.

(3) For fixed γ_0 and m and fixed levels ℓ_k, the empirical distribution of the entrance states in D_k tends to deteriorate (as an approximation of the exact distribution) as k increases, due the fact that it is derived from the empirical distribution at the previous level, so the approximation error accumulates from level to level. As a result, the variance reduction from array-RQMC is expected to decrease with k. Likewise, if $\gamma_0 = p^m$ for fixed p, the performance may deteriorate as a function of m.

(4) In the case of multidimensional state spaces, good choices of the importance function for splitting and of the sorting function for array-RQMC are crucial, and are definitely non-trivial to obtain in general. This is already the primary difficulty for either splitting or array-RQMC applied alone [Booth and Hendricks 1984; Garvels et al. 2002; L'Ecuyer et al. 2005].

(5) We saw earlier that under a number of conditions and approximations, splitting efficiency is maximized by taking $(-\ln \gamma_0)/2$ levels and selecting the thresh-

olds so that $p_k = e^{-2}$ for all k. For RESTART, a careful analysis by Villén-Altamirano and Villén-Altamirano [2002] shows that the c_k's should be as small as possible instead and the authors recommend $p_k = 1/2$. These optimal values largely depend on what variant of the splitting is used. In general, it is preferable to have fewer levels (smaller p_k's) when there is no truncation (all the chains must reach the next level or A). The truncation (or roulette) techniques permit one to use a larger number of levels without increasing too much the total amount of work. In a fixed-effort context, the number of chains is rebalanced more frequently when there is a larger number of levels. This frequent rebalancing is expected to help the array-RQMC method, because the number of chains will stay closer to the number of points in the RQMC point set. Our empirical investigations indicate that the optimal number m of levels for array-RQMC with fixed splitting tends to be larger than $(-\ln \gamma_0)/2$, but this depends very much on the problem. Fortunately, crude approximations may suffice because the variance is often not very sensitive to small changes in m.

(6) RQMC has been proved to be asymptotically more effective than MC only when the integrand is a *smooth* function [Owen 1998; L'Ecuyer and Lemieux 2002]. But here, we estimate the probability p_k of reaching the next level by an average of indicator functions, for each k. Indicator functions are definitely not smooth, so it is unclear a priori if array-RQMC can bring any improvement, even asymptotically. Proofs that the variance converges faster with RQMC than with MC are available only for narrow special cases for classical RQMC; for array-RQMC combined with splitting, the task is certainly not easier. This is a case where empirical experiments have the last word so far.

In the next section, we see that despite these difficulties, the combination can still bring significant variance reductions, at least in some situations.

5. EXAMPLES

Most of the following examples are toy problems, used to illustrate and evaluate the performance of various splitting and importance sampling methods and their combination with RQMC and array-RQMC.

When using splitting, we compare the different methods in terms of the *variance per chain*, $V_n = n\mathrm{Var}[\hat{\gamma}_n]$, where n is the number of chains at each level, and the *work-normalized variance per chain*, $W_n = S_n\mathrm{Var}[\hat{\gamma}_n]$, where S_n is the expected total number of simulated steps of the n Markov chains. If S_n is seen as the computing cost of the estimator, then $1/W_n$ is the usual measure of *efficiency*. For standard MC without splitting, and for the splitting variants where the chains remain independent (e.g., fixed splitting without truncation and resplits), V_n and W_n do not depend on n. But for the other methods, we also want to see how V_n and W_n behave as a function of n. If the RQMC and array-RQMC are really effective, for example, then V_n and W_n should decrease with n. We make no direct comparison with standard MC without splitting or IS, because this direct MC approach is not viable for our examples. Here we are only interested in the *additional* efficiency gains obtained by the truncation and RQMC methods. To measure the efficiency, we could also have measured the computing effort by the

total CPU time instead of S_n. However, the latter depends very much on the programming, compiler, and computer.

In our first example, we compare multilevel splitting with the fixed-splitting and fixed-effort strategies. For fixed splitting, we selected the splitting factors c_k so that $c_k \approx 1/p_k$ for all k. For the remainder of our experiments we use only fixed effort, because fixed splitting does not combine well with RQMC methods. We will test the work-saving truncation and resplit techniques of Section 2.7 (with $r_{k,1} = r_{k,2} = 1$, and $r_{k,j} = \hat{\gamma}_{(k-1)}^{-1/(k-1)}$ for $j \geq 3$, at all stages k), as well as the combination of splitting with RQMC methods. For all the splitting methods, the variance was estimated by making $R = 100$ independent replications of the entire procedure and taking the sample variance of the R estimators of the mean.

For classical RQMC, the point sets were infinite-dimensional Korobov lattice rules taken from Table 1 of L'Ecuyer and Lemieux [2000], to which we applied a random shift modulo 1 followed by the baker's transformation [Hickernell 2002]. This transformation maps each coordinate u to $2u$ if $u < 1/2$ and to $2(1-u)$ if $u \geq 1/2$; it produces a point set with "locally antithetic" properties [Hickernell 2002]. This method will be denoted *Classical-Korobov-Baker* in our tables. Again, we use infinite-dimensional point sets because the number of steps of each chain is random and unbounded.

For array-RQMC, we need to define a sorting function v, whose role is to distinguish the different states at any given step j of the chains within a stage k. One possibility is to just take v equal to h and this is what we do for our examples, even though there may be better choices. We use two different types of point sets for array-RQMC: (a) a two-dimensional Korobov lattice rule whose number of points n is a power of two and the parameter a is an odd integer such that a/n is near the golden ratio, with its first coordinate skipped, randomized by a random shift modulo 1 followed by the baker's transformation (we label this method *Array-Korobov-Baker* in the tables); (b) the first n points of a Sobol sequence randomized by a left (upper triangular) matrix scrambling followed by a random digital shift [L'Ecuyer and Lemieux 2002; Owen 2003], where the points are enumerated by order of their Gray code (*Array-Sobol*).

In our last set of examples, in Section 5.3, we will explain why splitting is not appropriate for certain reliability models and will experiment with the combination of IS with RQMC methods.

5.1 An Ornstein-Uhlenbeck Process

A continuous-time Ornstein-Uhlenbeck stochastic process $\{R(t),\, t \geq 0\}$ obeys the stochastic differential equation

$$dR(t) = a(b - R(t))dt + \sigma dW(t)$$

where $a > 0$, b, and $\sigma > 0$ are constants, and $\{W(t),\, t \geq 0\}$ is a standard Brownian motion [Taylor and Karlin 1998]. This model is also known as the Vasicek model for the evolution of short-term interest rates [Vasicek 1977]. In that context, b can be viewed as a long-term interest rate level toward which the process is attracted with strength $a(b - R(t))$. This process is *mean-reverting*, in the sense that it is attracted downward when it is high and attracted upward when it is low. The

constant σ indicates the strength of the noise.

Suppose the process is observed at times $t_j = j\delta$ for $j = 0, 1, \ldots$ and let $X_j = R(t_j)$. Let $A = (-\infty, b]$, $B = [\ell, \infty)$ for some constant ℓ, and $x_0 \geq b$. We want to estimate the probability that the process exceeds level ℓ at one of the observation times before it returns below b, when started from $R(0) = x_0$. Without loss of generality, we take $b = 0$. In terms of the transition function described earlier, we have

$$\varphi_j(x_j, U_j) = x_j e^{-a\delta} + \sigma \left(1 - e^{-2a\delta}/(2a)\right)^{1/2} \Phi^{-1}(U_j)$$

where Φ is the standard normal distribution function and U_j is uniformly distributed over $[0, 1)$.

A simple choice of importance function h in this case is the identity and we can combine it with equidistant thresholds. This choice is good enough to make splitting work, but it is certainly not optimal. In fact, the attraction of the process toward its mean is much stronger when it is farther away. So with equidistant thresholds, the probabilities p_k decrease with k, and it turns out that if $p_1 < 1/2$, for example, p_k becomes much too small when k is large. Therefore, if h is the identity, the thresholds should be placed closer to each other as k increases. Empirical experiments indicated that the following rule gives good results: first set $\ell_k = \ell\sqrt{k/m}$ for $k = 1, \ldots, m$, let k^* be the largest k for which $\ell_k < 2$, and reset $\ell_k = \ell_{k^*}(k/k^*)$ for $k = 1, \ldots, k^* - 1$. The latter is a correction that makes the first thresholds approximately equidistant. With this choice, our experiments indicated that the p_k's were roughly independent of k, except for the small values of k, where they varied a bit more. This is the choice we have used for the results reported in this section.

If we were considering the continuous-time process, the entrance distribution G_k at each level ℓ_k would be degenerate at ℓ_k. But because of the time discretization, the entrance distribution has positive support over the entire interval $[\ell_k, \infty)$, which means that a chain can cross an arbitrary number of thresholds in a single jump. One must be careful about this in the implementation. In this setting, the simulation starts from a fixed state only at the first level. The simulation in stage k determines $\hat{p}_k$ and the empirical distribution $\hat{G}_{k+1}$, which becomes the initial distribution for the simulation in stage $k + 1$. Our proofs of unbiasedness for the variants of multilevel splitting cover the case where the chain can cross several levels at a time.

Example 4. The results given here correspond to a discretely observed Ornstein-Uhlenbeck process with parameters $a = 0.1$, $b = 0$, $\sigma = 0.3$, $x_0 = 0.1$, and $\delta = 0.1$. We want to estimate the probability that the discrete-time process $\{X_j = R(t_j),\, j > 0\}$ exceeds $\ell = 4$ before getting below 0.

Savings from unbiased truncation. We first want to assess the gain provided by the truncation and resplit methods of Section 2.7, and see how this gain evolves when we change m and/or ℓ. Table I gives our estimates of V_n and W_n for some values of n, for MC with $m = 14$. In this (and other) tables, "Fixed splitting" and "Fixed effort" denote the corresponding multilevel splitting methods of Section 2.4, with MC and no truncation or resplits. The three "weight" lines correspond to the work-saving probabilistic truncation, periodic truncation, and tag-based truncation

methods described in Section 2.7, with weights and no resplits. The lines "Prob. resplit" and "Tag-based resplit" refer to probabilistic truncation with resplits and tag-based truncation with resplits, respectively (see Section 2.7).

To facilitate the reading, these estimates are rescaled as $\tilde{V}_n = 10^{16} \times \hat{V}_n$ and $\tilde{W}_n = 10^{13} \times \hat{W}_n$, where $\hat{V}_n$ and $\hat{W}_n$ are the empirical versions of V_n and W_n. We do not give $\tilde{V}_n$ for fixed splitting, because the number of chains at each step is random and not always equal to n. The table also gives the estimates (sample averages) $\hat{\gamma}_n$ and $\hat{S}_n$ of γ_0 and S_n for $n = 2^{16}$. The estimates of γ_0 are rescaled as $\tilde{\gamma}_n = \hat{\gamma}_n \times 10^8$. They agree quite well across the methods. The expected number of simulated steps, S_n, depends on whether or not we truncate and resplit, but is otherwise essentially independent of the method. The estimator $\hat{S}_n$ also appears to have small relative error.

The variance estimators, on the other hand, are quite noisy. This can be seen by looking at how $\tilde{V}_n$ behaves as a function of n for any given method. We also replicated some of the experiments a few times to see how the variance estimates varied, and they sometimes changed by more than 20%. To get a sense of the anticipated variability of the variance estimators, observe that if we assume normality of the $R = 100$ observations and use the chi-square distribution to compute a confidence interval on the variance (the classical procedure), with find that the error on the variance is (roughly) less than 10% with probability 0.95. But if we depart from the normality assumption, the error can be larger. Here, it seems to be somewhat larger, but no more than twice larger.

Without RQMC, fixed effort and fixed splitting turn out to have comparable efficiencies for this example. We nevertheless concentrate on fixed effort in what follows, because of its better compatibility with RQMC. Comparing the truncation methods, we expect truncation without resplits to give more variance than the version with resplits and also more variance than splitting without truncation. The results agree with this. The difference between the resplit versions and the standard splitting without truncation is lost in the noise. The truncation methods clearly increase the efficiency, by reducing the work. The versions without resplits reduce the work S_n by a factor of 4.3 while the versions with resplits reduce it by 3.5. Both versions reduce the work-normalized variance W_n approximately by a factor of 3.

Clearly, the benefit of truncation should increase with ℓ. For $\ell = 6$, for example, we have $\gamma_0 \approx 4.2 \times 10^{-18}$. By taking $m = 30$ and using our heuristic to determine the levels, we obtain p_k's of approximately the same size as with $\ell = 4$ and $m = 14$. Our numerical experiments in this case gave a reduction of S_n by a factor of 10 without resplits and a factor of 8 with resplits. Our estimator $\hat{W}_n$ was reduced by almost the same factors. Fixed effort and fixed splitting also had comparable efficiencies when no truncation and no RQMC was used.

We performed additional experiments to assess the impact of certain parameters such as the choice of levels ℓ_k (assuming that h is the identity function) and the parameters $r_{k,j}$ for the truncation. For $\ell = 4$ and $m = 14$, the empirical variance was only slightly smaller when the levels were chosen via our heuristic than with equidistant levels. For the truncation, taking $r_{k,1} = r_{k,2} = 1$ and $r_{k,j} = \hat{\gamma}_{(k-1)}^{-1/(k-1)}$ gave the best results. With $r_{k,2} > 1$, the variance is often significantly larger and,

Table I. Estimates of V_n and W_n for MC simulation with n chains, for $\ell = 4$ and $m = 14$ levels. The estimators are rescaled as $\tilde{V}_n = 10^{16} \times \hat{V}_n$, $\tilde{W}_n = 10^{13} \times \hat{W}_n$, and $\tilde{\gamma}_n = \hat{\gamma}_n \times 10^8$.

	$n = 2^{10}$		$n = 2^{12}$		$n = 2^{14}$		$n = 2^{16}$			
	$\tilde{V}_n$	$\tilde{W}_n$	$\tilde{V}_n$	$\tilde{W}_n$	$\tilde{V}_n$	$\tilde{W}_n$	$\tilde{V}_n$	$\tilde{W}_n$	$\tilde{\gamma}_n$	$\hat{S}_n$
Fixed splitting		197		203		183		219	1.59	9.59×10^7
Fixed effort	126	239	97	184	100	187	85	167	1.59	12.4×10^7
Prob. weight	126	56	107	56	138	61	146	65	1.58	2.92×10^7
Periodic weight	207	92	133	70	114	51	180	79	1.59	2.92×10^7
Tag-based weight	133	59	120	63	106	47	122	54	1.59	2.92×10^7
Prob. resplit	126	66	97	43	111	58	75	39	1.59	3.46×10^7
Tag-based resplit	111	58	97	42	90	47	117	62	1.59	3.46×10^7

more dangerously, is underestimated with a high probability because it contains a rare heavy contribution. We also tried $r_{k,j} = \hat{p}_{k-j} = R_{k,j}/N_{k-j-1}$ instead of $r_{k,j} = \hat{\gamma}_{(k-1)}^{-1/(k-1)}$ and none of the two choices dominated the other.

Classical RQMC. Table II shows our estimates of V_n and W_n for classical RQMC using the randomized infinite-dimensional Korobov point sets. The line "Splitting" represents fixed-effort splitting with classical RQMC and no truncation. Compared with MC, the classical RQMC reduces the variance roughly by a factor of 2 to 3 and requires about the same amount of work, regardless of n.

Table II. Estimates of V_n and W_n with classical RQMC, for $\ell = 4$ and $m = 14$ levels.

	$n = 1021$		$n = 4093$		$n = 16381$		$n = 65521$	
	$\tilde{V}_n$	$\tilde{W}_n$	$\tilde{V}_n$	$\tilde{W}_n$	$\tilde{V}_n$	$\tilde{W}_n$	$\tilde{V}_n$	$\tilde{W}_n$
Splitting	46	88	33	63	28	54	30	57
Prob. weight	80	36	77	34	86	38	96	43
Periodic weight	53	24	56	25	67	30	46	20
Tag-based weight	72	32	105	47	57	25	61	27

Array-RQMC. Table III gives our variance estimates for the combination of splitting with array-RQMC, using randomized Sobol' nets. For the sorting function, we simply use $v(x) = h(x) = x$. With truncation, the variance is about 12 times smaller with the resplits than without. One of the reasons is probably that without resplits, the chains have different weights and the sorting function does not take that into account. The truncation provides the same work reduction factor as for MC. The combination of array-RQMC with truncation and resplits provides a significant efficiency improvement, by a factor of about 200 to 250 for $n = 2^{16}$. This factor increases with n, which means that the variance decreases at a slightly faster rate than for MC. We performed a similar experiment with Korobov lattices, with the baker's transformation and a random shift, and the results were similar.

Changing m and ℓ. Table IV reports some experiments on the sensitivity to the choice of m, for $\ell = 4$ and $n = 2^{14}$. No truncation and resplit was used in the splitting variants. We see that reducing m decreases the simulation effort, but also increases the variance, and that both $m = 14$ and $m = 21$ appear like a good choices

Table III. Estimates of V_n and W_n for array-RQMC with Sobol' nets, for $\ell = 4$ and $m = 14$ levels.

	$n = 2^{10}$		$n = 2^{12}$		$n = 2^{14}$		$n = 2^{16}$	
	$\tilde{V}_n$	$\tilde{W}_n$	$\tilde{V}_n$	$\tilde{W}_n$	$\tilde{V}_n$	$\tilde{W}_n$	$\tilde{V}_n$	$\tilde{W}_n$
Splitting, with MC	126	239	97	184	100	187	85	167
Splitting	7.6	14	4.4	8.3	2.1	3.9	0.9	1.7
Prob. weight	30	13	23	10	30	13	25	11
Periodic weight	38	17	19	9	23	10	18	8
Tag-based weight	42	19	18	8	24	10	19	9
Prob. resplit	9.1	4.8	4.8	2.6	2.2	1.2	1.1	0.6
Tag-based resplit	11.0	5.9	4.1	2.2	2.6	1.4	1.3	0.7

in this case (in terms of W_n). If we assume that $\gamma_0 = p^m$ for some p, recalling that $\gamma_0 \approx 1.6 \times 10^{-8}$, we get

$p \approx 0.28$ for $m = 14$ and $p \approx 0.43$ for $m = 21$. These values are larger than the optimal value $p = e^{-2} \approx 0.135$ given in Section 2.3 under a simplified setting.

Table IV. Sensitivity of V_n ad W_n to the choice of m, for $\ell = 4$ and $n = 2^{14}$.

	$m = 7$		$m = 14$		$m = 21$	
	$S_n \approx 1.90 \times 10^7$		$S_n \approx 3.11 \times 10^7$		$S_n \approx 3.71 \times 10^7$	
	$\tilde{V}_n$	$\tilde{W}_n$	$\tilde{V}_n$	$\tilde{W}_n$	$\tilde{V}_n$	$\tilde{W}_n$
Splitting with MC	247	287	100	187	78	176
Classical RQMC	80	92	28	54	20	46
Array-RQMC-Sobol	7.5	8.7	2.1	3.9	1.8	4.0

5.2 A Tandem Queue

Example 5. We return to Example 3. As in Parekh and Walrand [1989], Glasserman et al. [1999], and Garvels [2000], among others, we consider an open tandem Jackson queueing network with two queues. The arrival rate at the first queue is $\lambda = 1$ while the mean service time is $\rho_i = 1/\mu_i$ at queue i, for $i = 1, 2$. The events are the arrivals and service completions (at any queue) and $X_j = (X_{1,j}, X_{2,j})$ is the number of customers in each of the two queues immediately after the jth event. We have $A = \{(0, 0)\}$, the state where the system is empty, and $B = \{(x_1, x_2) : x_2 \geq \ell\}$ for some fixed threshold ℓ, i.e., B is the set of states for which the length of the second queue is at least ℓ.

The choice of importance function h is a key issue for this example, especially when the bottleneck is at the first queue, i.e., if $\rho_1 > \rho_2$ [Glasserman et al. 1998; Garvels et al. 2002]. We shall concentrate on the case $\rho_1 < \rho_2$. We consider the following choices of h:

$$h_1(x_1, x_2) = x_2; \tag{8}$$

$$h_2(x_1, x_2) = (x_2 + \min(0, x_2 + x_1 - \ell))/2; \tag{9}$$

$$h_3(x_1, x_2) = x_2 + \min(x_1, \ell - x_2 - 1) \times (1 - x_2/\ell). \tag{10}$$

The function h_1 is a naive choice based on the idea that the set B is defined in terms of x_2 only. The second choice, h_2, counts ℓ minus half the minimal number

of steps required to reach B from the current state. (To reach B, we need at least $\ell - \min(0, x_2 + x_1 - \ell)$ arrivals at the first queue and $\ell - x_2$ transfers to the second queue.) The third choice, h_3, is adapted from Villén-Altamirano [2006], who recommends $h(x_1, x_2) = x_2 + x_1$ when $\rho_1 < \rho_2$. This h was modified as follows. We define $h_3(x) = x_1 + x_2$ when $x_1 + x_2 \le \ell - 1$ and $h_3(x) = \ell$ when $x_2 \ge \ell$. In between, i.e., in the area where $\ell - x_1 - 1 \le x_2 \le \ell$, we interpolate linearly in x_2 for any fixed x_1. This gives the function in (10).

For the sorting function, we simply take $v = h$.

We did a numerical experiment with $\rho_1 = 1/4$, $\rho_2 = 1/2$, and $\ell = 30$, with our three choices of h. With h_1, $\hat{V}_n$ and $\hat{W}_n$ were significantly higher than for h_2 and h_3, and none of the RQMC method provided any clear variance reduction, so we discard h_1. Tables V and VI summarize the results for h_2 and h_3, in the same format as for the previous tables. The estimators of V_n and W_n are rescaled as $\tilde{V}_n = 10^{18} \times \hat{V}_n$ and $\tilde{W}_n = 10^{15} \times \hat{W}_n$. The estimate of γ_0 is in the interval $1.29 \times 10^{-9} \pm 10^{-11}$ for all the methods.

Table V. Estimates of V_n and W_n for splitting using h_2, for the tandem queue with $\ell = 30$, and $m = 15$ levels. The estimators are rescaled as $\tilde{V}_n = 10^{18} \times \hat{V}_n$ and $\tilde{W}_n = 10^{15} \times \hat{W}_n$.

	$n = 2^{10}$		$n = 2^{12}$		$n = 2^{14}$		$n = 2^{16}$	
	$\tilde{V}_n$	$\tilde{W}_n$	$\tilde{V}_n$	$\tilde{W}_n$	$\tilde{V}_n$	$\tilde{W}_n$	$\tilde{V}_n$	$\tilde{W}_n$
Splitting, no-RQMC	109	120	89	98	124	137	113	125
Prob. weight	178	67	99	37	119	45	123	47
Periodic weight	99	37	118	45	107	41	119	45
Tag-based weight	126	48	133	50	109	41	109	41
Prob. resplit	104	45	138	60	96	42	111	48
Tag-based resplit	99	43	116	50	102	44	84	36
Classical RQMC	39	43	45	47	54	59	56	61
Array-Sobol	14	16	19	20	25	27	22	24
Array-Korobov-Baker	26	28	24	26	17	18	21	23
Array-Sobol, Prob. weight	28	11	31	12	27	10	28	11
Array-Korobov-Baker, Prob. weight	31	12	31	12	21	8.0	10	7.7
Array-Sobol, Periodic weight	29	11	43	16	28	10	23	8.7
Array-Korobov-Baker, Periodic weight	33	13	23	8.8	20	7.5	28	11
Array-Sobol, Tag-based weight	31	12	27	10	34	13	24	9.0
Array-Korobov-Baker, Tag-based weight	40	15	33	13	40	15	32	12
Array-Sobol, Prob. resplit	24	10	25	11	19	8.2	14	5.9
Array-Korobov-Baker, Prob. resplit	21	9.2	22	9.5	18	7.6	20	8.6
Array-Sobol, Tag-based resplit	21	9.2	60	26	17	7.4	18	7.8
Array-Korobov-Baker, Tag-based resplit	17	7.3	23	9.9	16	7.0	17	7.5

Without RQMC, h_3 gives estimators with almost the same variance as h_2 (just a little better). The truncation and resplit methods improve the efficiency (roughly) by a factor of 3, as in the previous example. We have slightly more variance reduction with the resplits than without, but also slightly more work, and the efficiency remains about the same.

Classical RQMC improves the V_n and W_n approximately by a factor of 2 with h_2 and 2.5 with h_3.

Table VI. Estimates of V_n and W_n for splitting using h_3, for the tandem queue with $\ell = 30$, and $m = 15$ levels.

	$n = 2^{10}$		$n = 2^{12}$		$n = 2^{14}$		$n = 2^{16}$	
	$\tilde{V}_n$	$\tilde{W}_n$	$\tilde{V}_n$	$\tilde{W}_n$	$\tilde{V}_n$	$\tilde{W}_n$	$\tilde{V}_n$	$\tilde{W}_n$
Splitting, no-RQMC	93	103	110	121	93	102	107	118
Prob. weight	90	34	93	35	94	36	109	41
Periodic weight	147	55	97	37	130	49	82	31
Tag-based weight	99	37	94	35	105	40	108	41
Prob. resplit	128	56	94	41	93	40	80	35
Tag-based resplit	84	36	100	44	86	37	100	44
Classical RQMC	49	55	36	39	39	43	40	44
Array-Sobol	19	21	14	15	7.6	8.4	3.9	4.3
Array-Korobov-Baker	13	14	10	11	6.2	6.8	3.7	4.1
Array-Sobol, Prob. weight	24	9.3	18	6.8	18	7.0	13	5.0
Array-Korobov-Baker, Prob. weight	26	10	19	7.2	13	4.8	12	4.9
Array-Sobol, Periodic weight	26	9.9	18	6.9	15	5.7	13	5.1
Array-Korobov-Baker, Periodic weight	18	6.7	14	9.1	14	5.2	14	5.4
Array-Sobol, Tag-based weight	24	9.1	21	7.9	19	7.3	13	4.8
Array-Korobov-Baker, Tag-based weight	17	6.4	20	7.7	18	6.9	14	5.3
Array-Sobol, Prob. resplit	22	9.6	11	4.7	6.3	2.8	5.4	2.4
Array-Korobov-Baker, Prob. resplit	14	6.2	8.0	3.5	6.0	2.6	2.7	1.2
Array-Sobol, Tag-based resplit	21	8.9	12	5.0	6.6	2.9	4.8	2.1
Array-Korobov-Baker, Tag-based resplit	15	6.6	11	4.7	4.1	1.8	3.4	1.5

The array-RQMC methods behave differently for h_2 and h_3. With h_2, the efficiency is improved roughly by a factor of 5 without truncation, a factor of 12 with truncation, and a factor of 15 with truncation and resplits. With h_3, these factors are of the same order for $n = 2^{10}$, but for $n = 2^{16}$ they are approximately 25, 20, and 50 to 70 (50 for Sobol' and 70 for Korobov), respectively. With h_3, the array-RQMC method without truncation turns out to be very effective and the gain improves rapidly with n. Its combination with truncation and resplit is even more effective. Without the resplits, however, the truncation increases the variance so much that it reduces the overall efficiency. With the resplits, Korobov is slightly better than Sobol' in all eight cases with h_3. The best empirical efficiency (1.2 for $n = 2^{16}$) is obtained by array-RQMC with a Korobov point set and probabilistic truncation with resplits. Its overall efficiency improvement factor (on top of what is obtained by splitting alone with h_3) is nearly 100. Choosing a good sorting function together with a good importance function is a key factor to make the array-RQMC method really effective. For the results reported so far, we always took $v = h$. To get a idea of whether the improvement observed with h_3 is due mainly to the better choice of h or to the better choice of v, we made additional experiments in which we used $h = h_2$ with $v = h_3$, and vice-versa, for the Array-Sobol and Array-Korobov-Baker methods. We observed a greater dependence on the sorting function v than on the importance function h (between h_2 and h_3). In comparison with $h = v = h_3$, using $h = h_2$ and $v = h_3$ gave a slight increase of empirical variance (by less than 50%) for $k = 16$ and no significant change for smaller values of k, whereas with $h = h_3$ and $v = h_2$, the empirical variance increased by factors

ranging (roughly) from 2 to 6. So in this case, switching between h_2 and h_3 has more impact for v than for h.

5.3 Highly Reliable Multicomponent Systems

We return to Example 1, for which we want to estimate the probability γ_0 that the system fails before it returns to its initial state, with all the components operational. Here we consider IS, rather than splitting, in combination with RQMC. We also discuss splitting and its limitations for this type of application.

As in Shahabuddin [1994b], we assume that the failure rate of type-i components when in state x is $\lambda_i(x) = a_i(x)\varepsilon^{b_i(x)}$, where $a_i(x)$ is a strictly positive real number when at least one type-i component is operational in state x and $b_i(x)$ is a strictly positive integer, both independent of the *rarity parameter* ε, which satisfies $0 < \varepsilon \ll 1$. In full generality, failure propagation is allowed: from state x, there is a probability $p_i(x, y)$ that the failure of a type-i component directly drives the system to state y, in which there could be additional component failures. Thus, the net jump rate from x to y is $\lambda(x, y) = \sum_{i=1}^{c} \lambda_i(x) p_i(x, y) = O(\varepsilon)$. Similarly, the repair rate from state x to state y is $\mu(x, y)$ (with possible grouped repairs), where $\mu(x, y)$ does not depend on ε (i.e., repairs are not rare events when they are possible). We also assume that at least one repairman is active whenever a component is failed. We want to estimate $\gamma_0 = \mathbb{P}[\tau_B < \tau_A]$, defined in Example 1. For this, it suffices to simulate the embedded discrete-time Markov chain $\{X_j, j \geq 0\}$. We denote its transition probabilities by $P(x, y) = \mathbb{P}[X_j = y \mid X_{j-1} = x]$, for $x, y \in \mathcal{X}$, with $P(x, y) = \lambda(x, y)/q(x)$ if the transition from x to y results from a failure, and $P(x, y) = \mu(x, y)/q(x)$ if it is a repair, $q(x) = \sum_{y \in \mathcal{X}} (\lambda(x, y) + \mu(x, y))$ being the total flow rate out of x.

Various heuristics have been proposed to select a change of measure for IS in this setting [Nakayama 1996; Cancela et al. 2002]. The most robust approach with respect to rarity is the *balanced failure biasing* (BFB) scheme developed by Shahabuddin [1994b]; it has bounded relative error when $\varepsilon \to 0$. See Nakayama [1996], Tuffin [1999], Tuffin [2004], and Cancela et al. [2005] for discussions of robustness. BFB replaces the $P(x, y)$'s by new transition probabilities $P'(x, y)$ 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

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

For $x \in B$, we take $P'(x, y) = P(x, y)$. This change of measure increases the probability of failure to at least ρ when the system is up, so a failure transition is no longer a rare event.

A second approach is *inverse failure biasing* (IFB) [Papadopoulos 1998], which

follows the idea of *exponential twisting* in queueing networks; it inverts the probabilities of the sets $F(x)$ and $R(x)$, taking uniform probabilities in those subsets. This gives

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

Though this scheme does not verify robustness properties such as *bounded relative error* (BRE) in full generality [Cancela et al. 2002], because the most likely paths to failure may involve some repairs and thus become rare under IFB, it does for our numerical examples. It also performs better than BFB in many situations, see Example 6 below.

With IS, the contribution of a chain to the estimator is the likelihood ratio

$$L = \prod_{j=1}^{\tau_B} \frac{P(X_{j-1}, X_j)}{P'(X_{j-1}, X_j)}$$

multiplied by the indicator $\mathbb{I}[\tau_B < \tau_A]$. The IS estimator of γ_0 is the average value of these contributions over the n Markov chains that are simulated.

We will try IS with MC and with array-RQMC for a few examples. For the choice of the sorting function v for array-RQMC, it seems reasonable to take some kind of forecast of the final value of $L \cdot \mathbb{I}[\tau_B < \tau_A]$. Here, as a crude (easy to compute) estimate, we take $v(x)$ as the largest value of L that can be obtained, over all sample paths that reach B, given the current state x and the likelihood ratio cumulated so far. Sometimes, we will also consider a variant that looks at the number of failed components as well.

Example 6. We start with the simple case where $c = 1$, so the system has n_1 components all of the same type and the state X_j is the number of failed components. We take $\varepsilon = 0.05$, $\mu(x,y) = \mu_1 = 1$, $\lambda_1(x) = \lambda_1 = 0.001$, $n_1 = 10$, $\rho = 0.5$, and B is the set where fewer than two components are operational. For this example, we have $\gamma_0 \approx 9.0 \times 10^{-24}$ and the variance per run for MC with IS is approximately 1.99×10^{-44} for BFB and 1.45×10^{-48} for IFB. So the variance is about 10,000 times smaller with IFB.

For this particular example, there is another change of measure that guarantees a sharp bound on the likelihood ratio. When in state x, the probability that the next event is a failure is $q_x = p_{\mathrm{F}}(x) = x\lambda_1/(x\lambda_1 + (n_1 - x)\mu_1)$ for $x > 0$, and $q_0 = 1$. Select $q'_1 > q_1$ and define recursively $q'_{x+1} = 1 - (1 - q_{x+1})(q_x/q'_x)$. These q'_x's suffice to simulate the chain. We then have $q'_x > q_x$ for all $x > 0$. This also gives $(q_x/q'_x)(1 - q_{x+1})/(1 - q'_{x+1}) = 1$, which means that all cycles in the sample path contribute nothing to the likelihood ratio (such a change of measure is much more difficult to construct in the multidimensional case). As a result, whenever we reach B, we have

$$L = \prod_{i=1}^{n_1 - 2} (q_x/q'_x),$$

a constant. Note that γ_0 is equal to this L multiplied by the probability γ'_0 of reaching B under IS. To increase γ'_0, i.e., make the constant L as small as possible, we would choose q'_1 as large as possible, i.e., $q'_1 = 1$. This algorithm will be denoted by NLC (for *no loop contribution*). On our example, it gives an (empirical) variance per run of approximately 2.72×10^{-48}, which is about the same as for IFB.

Table VII gives the (empirical) additional variance reduction factor obtained when we combine IS with RQMC; that is, the variance with IS + MC divided by the variance with the same IS technique + the named RQMC method (the larger the better, in contrast with the previous examples). The sorting function v_1 only looks at the largest possible likelihood ratio as explained earlier, whereas v_2 uses a lexicographic order: it first looks at the number of failed components, and the chains that have the same number are sorted according to the current value of the likelihood ratio. The RQMC methods use n points (or almost). The number of independent replications used to estimate the variance (in this example and those that follow) is $R = 256$.

To compare the no-RQMC IS methods between themselves, we can look at the variance reduction factor of each method compared with BFB. This factor is approximately 14000 for IFB and 7300 for NLC

(it does not depend on v).

Table VII. Empirical variance reduction factors of RQMC with respect to MC, for Example 6, with n points and $R = 256$ replications (for classical RQMC, n is the largest prime number smaller than the given value).

IS	v		$n = 2^{10}$	$n = 2^{12}$	$n = 2^{14}$	$n = 2^{16}$	$n = 2^{18}$
BFB	v_1	Classical-Korobov-Baker	1.0	2.6	2.8	5.2	9.6
		Array-Sobol	2.8E+7	2.6E+10	9.4E+13	3.4E+17	1.5E+21
		Array-Korobov-Baker	11	21	53	17	134
BFB	v_2	Classical-Korobov-Baker	1.0	2.2	3.1	4.5	8.2
		Array-Sobol	2.4E+7	1.2E+11	1.2E+14	5.4E+17	2.3E+21
		Array-Korobov-Baker	10	25	53	19	132
IFB	v_2	Classical-Korobov-Baker	19	23	43	134	118
		Array-Sobol	14	26	65	124	420
		Array-Korobov-Baker	16	23	67	17	122
NLC	v_2	Classical-Korobov-Baker	5.8	9.6	23	33	50
		Array-Sobol	5.2	12	21	33	68
		Array-Korobov-Baker	4.4	12	20	12	39

We see that RQMC reduces the variance by significant factors and the reduction also increases with n. The combination of BFB with Array-Sobol gives surprisingly good results, with practically zero variance when $n = 2^{18}$. At first, we thought it was a programming error of some sort, but then realized that this is due to a constructive interaction between the randomized Sobol' point set and the modified probabilities $P'(x, y)$. By taking $\rho = 0.5$, for any state where both a failure and a repair are possible, the probability of each is 0.5. But for a two-dimensional randomized Sobol' point set with 2^k points, for any interval of the form $[a_1 2^{-k+1}, a_2 2^{-k+1})$ where a_1 and a_2 are integers such that $0 \le a_1 < a_2 \le 2^{k-1}$, *exactly half* of the

points whose first coordinate is in this interval have their second coordinate smaller than 0.5. Because of this, after the first failure, a second failure will occur immediately for exactly half of the chains, and a repair for the other half. Then, exactly half of the chains with two failures will have another failure for their next transition, and so on. At most steps and for most states, exactly half of the chains in that state have a failure and the other half have a repair. As long as this happens, the empirical distribution of the states reproduces exactly the theoretical distribution, so the variance is almost nil. This does not happen for the Korobov point sets, since they do not have this property, but Array-Korobov-Baker nevertheless gives a substantial improvement. With classical RQMC, there is a good improvement for IFB but not quite as good for BFB. The choice of v is irrelevant for classical RQMC, so the difference between v_1 and v_2 with BFB is only noise. It gives a sense of the uncertainty in the variance estimate: the observed difference sometimes goes up to almost 20%.

We also considered splitting for this example, but met the following difficulty: When $q_x = x\lambda_1/(x\lambda_1 + (n_1 - x)\mu_1)$ is too small, it is just impossible to set the thresholds close enough to each other for the splitting to be effective. For $n_1 = 10$, if $h(x) = x$, the only possible thresholds are $2, 3, \ldots, 8$. With these thresholds and our selected parameters, the probability of reaching level 8 given that we have reached level 7 is $p_8 \approx 0.0004$, which is quite small. Too few of the chains started at a given level will make it to the next level, especially for the highest levels. Splitting does not work well in this case (the relative error remains very large). To view the difficulty from another angle, suppose we insist that the splitting factor c_k never exceeds 100 (which is already quite large). If $n_1 = 10$ and $\mu_1 = 1$, then we must have $\lambda_1 \approx 0.024$. If $n_1 = 40$ instead, then we must have $\lambda_1 \approx 0.12$.

Example 7. Consider now a system with $c = 3$ component types, $\mu_i = 1$ for all i, $\lambda_1 = \varepsilon$, $\lambda_2 = 1.5\varepsilon$, and $\lambda_3 = 2\varepsilon^2$, where μ_i and λ_i are the repair and failure rates of component type i. Suppose the system is down when fewer than two components of anyone type are operational. We assume that n_i is the same for all i, for $1 \le i \le c$, and consider the three parameter sets given in Table VIII. The sorting function v for array-RQMC is v_1, as defined earlier. The table also gives the variance per run V_n for BFB and IFB, without RQMC. As for the previous example, we see that IFB is more effective than BFB. The two IS methods are also much less effective to reduce the variance when $n_i = 12$ than for the smaller values of n_i (recall that without IS, the variance per run is $\gamma_0(1 - \gamma_0) \approx \gamma_0$). This is due to the fact that these IS methods are not adapted to this case. Indeed, BFB and IFB schemes have actually been designed to cope with the case where individual failure rates are small (i.e., when $\varepsilon \to 0$) and not for the case where the number of components is large.

Table VIII. Parameters and MC variance estimates for the 3-dimensional example.

n_i	ε	γ_0	V_n with BFB	V_n with IFB
3	0.001	2.6×10^{-3}	6.21×10^{-5}	2.83×10^{-5}
6	0.01	1.8×10^{-7}	6.17×10^{-11}	9.42×10^{-12}
12	0.1	6.0×10^{-8}	7.48×10^{-8}	2.71×10^{-8}

Table IX. Empirical variance reduction factors of RQMC with respect to MC, for the 3-dimensional example, with n points and $R = 256$ replications (for classical RQMC, n is the largest prime number smaller than the given value).

	n_i	IS	$n = 2^{10}$	$n = 2^{12}$	$n = 2^{14}$	$n = 2^{16}$	$n = 2^{18}$
3	BFB	Classical-RQMC-Baker	73	28	40	107	7910
		Array-Sobol	24	74	250	720	3200
		Array-Korobov-Baker	18	57	467	123	6120
3	IFB	Classical-RQMC-Baker	56	10	38	32	287
		Array-Sobol	23	51	71	241	613
		Array-Korobov-Baker	7	53	146	460	1440
6	BFB	Classical-RQMC-Baker	1.0	1.0	1.4	1.9	3.9
		Array-Sobol	1.1	1.4	3.2	6.2	20.0
		Array-Korobov-Baker	1.3	2.0	3.9	6.7	35.0
6	IFB	Classical-RQMC-Baker	1.9	0.5	1.8	3.0	2.7
		Array-Sobol	1.2	2.4	2.5	4.0	4.8
		Array-Korobov-Baker	2.5	2.9	2.6	4.4	5.3
12	BFB	Classical-RQMC-Baker	1.0	110	7	23	9
		Array-Sobol	140	21	150	14	0.1
		Array-Korobov-Baker	1400	500	265	80	5
12	IFB	Classical-RQMC-Baker	7850	182	6	0.01	1.3
		Array-Sobol	3300	460	70	34	10
		Array-Korobov-Baker	560	120	22	50	14

Table X. Estimated values $\hat{\gamma}_n$ by RQMC for the 3-dimensional example, for $n_i = 12$, with n points and $m = 256$ replications.

n_i	IS		$n = 2^{10}$	$n = 2^{12}$	$n = 2^{14}$	$n = 2^{16}$	$n = 2^{18}$
12	BFB	Classical-RQMC	5.4E-7	5.3E-8	8.8E-8	5.0E-8	5.5E-8
		Array-Sobol	7.3E-8	9.2E-8	2.8E-8	7.1E-8	1.7E-7
		Array-Korobov-Baker	2.2E-8	1.9E-8	3.1E-8	4.1E-8	6.3E-8
12	IFB	Classical-RQMC	1.1E-8	4.0E-8	5.7E-8	4.9E-7	7.0E-8
		Array-Sobol	1.1E-8	2.1E-8	3.2E-8	3.8E-8	5.0E-8
		Array-Korobov-Baker	3.6E-8	2.3E-8	5.1E-8	3.3E-8	4.6E-8

Variance reduction factors for RQMC versus MC (the MC variance divided by the RQMC variance, for the corresponding IS method), for both BFB and IFB, are given in Table IX. The expected number of simulation steps per chain is the same for all the methods. However, the array-RQMC methods require more CPU time because of the need to compute the sorting function and sort the chains at each step. The increase in CPU time depends very much on the implementation and computer and it could be more than a factor of 10 when the sorting function takes much more time to compute than simulating the next step of a chain.

For $n_i = 3$, the RQMC methods perform very well and reduce the variance by a (large) factor that increases with n. Classical RQMC provides reductions similar to array-RQMC in this case. This is probably due to the small effective dimension (the small number of simulation steps required to reach failure).

When the number of components increases ($n_i = 6$ or 12), on the other hand, RQMC appears much less effective to reduce the variance. For $n_i = 12$, since the IS methods provides almost no variance reduction, the relative error on γ_0 remains high and we expect the relative error on the variance estimators to be even higher. This is what explains the strange (noisy) results in Table IX, where the empirical variance reduction factors of RQMC decrease with n: For small n, the RQMC variance estimates are so noisy that they often grossly underestimate the true variance, whence the large numbers in the table. What happens is that, even when applying the IS methods, there are still in this case important events with a large contribution to the estimator (e.g., they give a large likelihood ratio) but which occur very rarely. When none of these rare events occurs, the mean and variance are both grossly underestimated. Table X shows the estimates of γ_0 for the RQMC methods for $n_i = 12$. When the mean estimate is much smaller than 6×10^{-8}, this suggests that the corresponding variance reduction factor in Table IX is grossly overestimated. For classical RQMC with IFB, for instance, we see that the mean is grossly underestimated for $n = 2^{10}$ and grossly overestimated for $n = 2^{16}$, so the corresponding empirical variance reduction factors of 7850 and 0.01 given in the table are probably gross overestimates and underestimates, respectively, of the true factors.

To illustrate this phenomenon, we made 10 independent MC simulations of 10^6 paths each, with BFB. Among those 10 runs, the smallest value of the mean estimator was 1.33×10^{-8}, with an empirical variance per run of 1.99×10^{-11}, and the largest was 4.50×10^{-6}, with an empirical variance per run of 1.98×10^{-5}. Increasing the number R of independent replications to better estimate the mean and the variance is not really a viable solution in this context. The rare-event problem must also be addressed *before* RQMC methods can be of any use; RQMC methods are not designed to solve it. To handle the difficulty, one would need to design a different IS technique or perhaps a clever combination of IS with splitting. L'Ecuyer and Tuffin [2006] explore the use of weight windows to improve the IS estimators for this example.

Example 8. We now consider a simplified version of an example taken from Shahabuddin [1994a] that better fits the BFB framework. A system is comprised of two sets of processors with two processors per set and two sets of disk controllers with two controllers per set. We also have six clusters of disks with four disks per cluster. Failure rates of processors, controllers, and disks are 5×10^{-5}, 2×10^{-5} and 2×10^{-5}, respectively. The repair rate is 1 for each type of component.

There are two modes of failures occurring with equal probability. In each disk cluster, data is replicated, which means that one disk can fail without affecting the system. The system is operational if all data is accessible from both processor types, meaning that at least one processor of each type, one controller of each set, and three disks of each cluster need to be operational. This can be modeled by $c = 10$ different types of components to differentiate between the different sets of the same kind of component.

We made some numerical experiments using BFB and IFB, with MC, classical RQMC and array-RQMC. For BFB, we used $\rho = 0.5$. The exact probability in this case is approximately 5.55×10^{-5}. Table XI gives the variance reduction

factor of each RQMC method with respect to MC (with the same IS method). We see that classical RQMC performs much better than array-RQMC and provides a significant improvement compared with MC. This improvement can be explained by the small effective dimension: In this context of multiple component types with a small number of component failures required for the system failure, the Markov chain typically needs only a few steps to reach its stopping time τ, under IS, so its simulation only requires a few random numbers. For the same reason, splitting (without IS) would not work for this example, because there is no way to set the thresholds so that the p_k's are large enough. Thus, classical RQMC combines very well with the IS simulation of the types of highly reliable Markovian systems studied by Shahabuddin [1994b], whereas splitting hardly applies.

Array-RQMC provides a very modest variance reduction with Korobov point sets, and (surprisingly) a variance increase with the Sobol' point sets. Even more surprisingly, the variance with the Sobol' point sets *increases* with n. We have no clear explanation for this phenomenon at the moment, but it could be that the structure of the Sobol' point set interacts with the problem in a way that reduces the frequency of failures. We must underline that the choice of sorting function v is very important for the array-RQMC method to work well. So we may be able to improve the results significantly by changing v. But here the state space has 10 dimensions and it seems difficult to find a good v; we chose $v = v_1$ and it performs rather poorly.

Table XI. Empirical variance reduction factors of RQMC with respect to MC, for the 10-dimensional example, with $n \approx 2^k$ points and $m = 256$ replications.

IS method		$n = 2^{10}$	$n = 2^{12}$	$n = 2^{14}$	$n = 2^{16}$	$n = 2^{18}$
BFB	Classical-rqmc-Baker	8	22	9	276	27
	Array-Sobol	0.3	0.1	0.03	0.01	0.003
	Array-Korobov-Baker	1.3	4.8	2.2	0.8	1.1
IFB	Classical-rqmc-Baker	11	40	181	96	11
	Array-Sobol	0.7	0.3	0.09	0.003	0.008
	Array-Korobov-Baker	2.1	4.0	5.7	1.8	3.1

6. CONCLUSION

We gave an overview of multilevel splitting for rare event simulation, proposed some improvements via truncation and resplits within each stage, and examined the combination with RQMC methods. Our numerical examples showed that when splitting applies and the importance function is well chosen, these improvements combined with the RQMC methods can reduce the variance and improve the efficiency by significant factors on top of those already obtained by the standard multilevel splitting alone.

Our empirical comparisons are based on somewhat noisy variance estimates (with up to 20% error) for the RQMC methods, as is usually the case. On the other hand, the empirical efficiency improvement factors of over 200 and 100 observed in Examples 4 and 5 for array-RQMC compared with splitting alone, as well as the

spectacular improvement observed in Example 6, are definitely much more important than this noise. When RQMC methods are used in practice, it is customary to make no more than about 10 independent replications. Then the variance estimates often have more than 20% error. When a large computing budget is available, one usually prefers to increase n instead of R, because this gives a better estimator of the mean.

The unbiased truncation methods with resplits did improve the efficiency of multilevel splitting in all our experiments, by factors ranging from 2 to 10. They increase the variance slightly but reduce the work by a larger factor. Without RQMC, the difference in performance between the variants was not very significant. With array-RQMC, however, the two variants that use resplits performed much better than those using weights. Between those two, we recommend the *probabilistic truncation with resplits*, because it is simpler and easier to implement. The performance of array-RQMC with splitting depends (sometimes strongly) on good choices of *both* h and v, so one must be very careful in selecting them. Unfortunately, good choices are not always easy to find. For this, adaptive methods that estimate the function γ along the way appear promising. Further work is needed in that direction.

We pointed out limitations of the splitting methodology. The method is not appropriate, for example, for the type of HRMS considered by Shahabuddin [1994b], because there is no way to define the levels close enough so that the probability of reaching the next level is reasonably large. For these types of systems, other techniques such as IS must be used to push the system toward the important rare events. We compared (via numerical examples) two ways of applying IS: the BFB of Shahabuddin [1994a] and the IFB of Papadopoulos [1998], and their combination with RQMC methods. On our examples, without RQMC, IFB performed much better than BFB, and the RQMC methods gave slightly more improvement over BFB than over IFB. Our last example illustrated the fact that classical RQMC is likely to perform better than array-RQMC if the state space has large dimension and the Markov chain only needs a few steps to reach the rare event under IS. We also found a situation where a synergetic effect between BFB and array-RQMC with Sobol' point sets produced an astonishing variance reduction, by a factor of over 10^{21}. We believe that simulation of Markov chains can sometimes be designed (by an appropriate change of measure) to exploit this type of effect. This is a good topic for further research.

ACKNOWLEDGMENTS

This work has been supported by NSERC-Canada grant No. ODGP0110050 and a Canada Research Chair to the first author, an NSERC-Canada scholarship to the second author, and EuroNGI Network of Excellence and "SurePath ACI Security" Project to the third author, and an FQRNT-INRIA Travel Grant to all three authors. We thank Tom Booth for sending us several insightful articles. The suggestions from an anonymous reviewer and the Editor Jim Wilson helped improving the paper.

REFERENCES

Andradóttir, S., Heyman, D. P., and Ott, T. J. 1995. On the choice of alternative measures in importance sampling with Markov chains. *Operations Research 43,* 3, 509–519.

Asmussen, S. 2002. Large deviations in rare events simulation: Examples, counterexamples, and alternatives. In *Monte Carlo and Quasi-Monte Carlo Methods 2000*, K.-T. Fang, F. J. Hickernell, and H. Niederreiter, Eds. Springer-Verlag, Berlin, 1–9.

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

Booth, T. E. 1985. Monte Carlo variance comparison for expected-value versus sampled splitting. *Nuclear Science and Engineering 89*, 305–309.

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

Booth, T. E. and Pederson, S. P. 1992. Unbiased combinations of nonanalog Monte Carlo techniques and fair games. *Nuclear Science and Engineering 110*, 254–261.

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

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

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

Cérou, F. and Guyader, A. 2005. Adaptive multilevel splitting for rare event analysis. Tech. Rep. 5710, INRIA. Oct.

Cérou, F., LeGland, F., Del Moral, P., and Lezaud, P. 2005. Limit theorems for the multilevel splitting algorithm in the simulation of rare events. In *Proceedings of the 2005 Winter Simulation Conference*, F. B. A. M. E. Kuhl, N. M. Steiger and J. A. Joines, Eds. 682–691.

Chang, C. S., Heidelberger, P., Juneja, S., and Shahabuddin, P. 1994. Effective bandwidth and fast simulation of ATM intree networks. *Performance Evaluation 20*, 45–65.

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

Ermakov, S. M. and Melas, V. B. 1995. *Design and Analysis of Simulation Experiments*. Kluwer Academic, Dordrecht, The Netherlands.

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

Garvels, M. J. J. 2000. The splitting method in rare event simulation. Ph.D. thesis, Faculty of mathematical Science, University of Twente, The Netherlands.

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

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

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

Glasserman, P., Heidelberger, P., Shahabuddin, P., and Zajic, T. 1998. A large deviations perspective on the efficiency of multilevel splitting. *IEEE Transactions on Automatic Control AC-43,* 12, 1666–1679.

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

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

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

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

Hammersley, J. M. and Handscomb, D. C. 1964. *Monte Carlo Methods*. Methuen, London.

Harris, T. 1963. *The Theory of Branching Processes*. Springer-Verlag, New York.

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

Heidelberger, P., Shahabuddin, P., and Nicola, V. F. 1994. Bounded relative error in estimating transient measures of highly dependable non-Markovian systems. *ACM Transactions on Modeling and Computer Simulation 4,* 2, 137–164.

Hickernell, F. J. 2002. Obtaining $o(n^{-2+\epsilon})$ convergence for lattice quadrature rules. In *Monte Carlo and Quasi-Monte Carlo Methods 2000*, K.-T. Fang, F. J. Hickernell, and H. Niederreiter, Eds. Springer-Verlag, Berlin, 274–289.

Juneja, S. and Shahabuddin, P. 2001. Fast simulation of Markov chains with small transition probabilities. *Management Science 47,* 4, 547–562.

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

Keilson, J. 1979. *Markov Chain Models—Rarity and Exponentiality*. Springer-Verlag, New York.

Lécot, C. and Tuffin, B. 2004. Quasi-Monte Carlo methods for estimating transient measures of discrete time Markov chains. In *Monte Carlo and Quasi-Monte Carlo Methods 2002*, H. Niederreiter, Ed. Springer-Verlag, Berlin, 329–343.

L'Ecuyer, P., Lécot, C., and Tuffin, B. 2005. A randomized quasi-Monte Carlo simulation method for Markov chains. submitted.

L'Ecuyer, P., Lécot, C., and Tuffin, B. 2006. Randomized quasi-Monte Carlo simulation of Markov chains with an ordered state space. In *Monte Carlo and Quasi-Monte Carlo Methods 2004*, H. Niederreiter and D. Talay, Eds. 331–342.

L'Ecuyer, P. and Lemieux, C. 2000. Variance reduction via lattice rules. *Management Science 46,* 9, 1214–1235.

L'Ecuyer, P. and Lemieux, C. 2002. Recent advances in randomized quasi-Monte Carlo methods. In *Modeling Uncertainty: An Examination of Stochastic Theory, Methods, and Applications*, M. Dror, P. L'Ecuyer, and F. Szidarovszky, Eds. Kluwer Academic, Boston, 419–474.

L'Ecuyer, P. and Tuffin, B. 2006. Splitting with weight windows to control the likelihood ratio in importance sampling. In *Proceedings of ValueTools 2006: International Conference on Performance Evaluation Methodologies and Tools*. ACM Publications, Pisa, Italy.

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

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

Nakayama, M. K. and Shahabuddin, P. 1998. Likelihood Ratio Derivative Estimation for Finite-Time Performance Measures in Generalized Semi-Markov Processes. *Management Science 44,* 10, 1426–1441.

Nicola, V. F., Shahabuddin, P., and Heidelberger, P. 1993. Techniques for fast simulation of highly dependable systems. In *Proceedings of the 2nd International Workshop on Performability Modelling of Computer and Communication Systems*.

Niederreiter, H. 1992. *Random Number Generation and Quasi-Monte Carlo Methods*. SIAM CBMS-NSF Regional Conference Series in Applied Mathematics, vol. 63. SIAM, Philadelphia.

Owen, A. B. 1998. Latin supercube sampling for very high-dimensional simulations. *ACM Transactions on Modeling and Computer Simulation 8,* 1, 71–102.

Owen, A. B. 2003. Variance with alternative scramblings of digital nets. *ACM Transactions on Modeling and Computer Simulation 13,* 4, 363–378.

Papadopoulos, C. 1998. A new technique for MTTF estimation in highly reliable Markovian systems. *Monte Carlo Methods and Applications 4,* 2, 95–112.

Parekh, S. and Walrand, J. 1989. A quick simulation method for excessive backlogs in networks of queues. *IEEE Transactions on Automatic Control AC-34*, 54–56.

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

Shahabuddin, P. 1994b. Importance sampling for the simulation of highly reliable markovian systems. *Management Science 40,* 3, 333–352.

Shahabuddin, P. 1995. Rare event simulation in stochastic models. In *Proceedings of the 1995 Winter simulation Conference*. IEEE Press, 178–185.

Shahabuddin, P., Nicola, V. F., Heidelberger, P., Goyal, A., and Glynn, P. W. 1988. Variance reduction in mean time to failure simulations. In *Proceedings of the 1988 Winter Simulation Conference*. IEEE Press, 491–499.

Taylor, H. M. and Karlin, S. 1998. *An Introduction to Stochastic Modeling*, third ed. Academic Press, San Diego.

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

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

Vasicek, O. 1977. An equilibrium characterization of the term structure. *Journal of Financial Economics 5*, 177–188.

Villén-Altamirano, J. 2006. Rare event RESTART simulation of two-stage networks. *European Journal of Operations Research*. to appear.

Villén-Altamirano, M., Martinez-Marrón, A., Gamo, J., and Fernández-Cuesta, F. 1994. Enhancement of the accelerated simulation method restart by considering multiple thresholds. In *Proceedings of the 14th International Teletraffic Congress*. Elsevier Science, 797–810.

Villén-Altamirano, M. and Villén-Altamirano, J. 1994. RESTART: A straightforward method for fast simulation of rare events. In *Proceedings of the 1994 Winter Simulation Conference*. IEEE Press, 282–289.

Villén-Altamirano, M. and Villén-Altamirano, J. 2002. Analysis of RESTART simulation: Theoretical basis and sensitivity study. *European Transactions on Telecommunications 13,* 4, 373–386.

Villén-Altamirano, M. and Villén-Altamirano, J. 2006. On the efficiency of RESTART for multidimensional systems. *ACM Transactions on Modeling and Computer Simulation 16,* 3, 251–279.

Wilson, J. R. 1983. Antithetic sampling with multivariate inputs. *American Journal of Mathematical and Management Sciences 3*, 121–144.

This version: August 24, 2006.