Path-ZVA: General, Efficient, and Automated Importance Sampling for Highly Reliable Markovian Systems

DANIËL REIJSBERGEN, University of Edinburgh, Scotland
PIETER-TJERK DE BOER and WERNER SCHEINHARDT, University of Twente, Netherlands
SANDEEP JUNEJA, Tata Institute of Fundamental Research, India

We introduce Path-ZVA: an efficient simulation technique for estimating the probability of reaching a rare goal state before a regeneration state in a (discrete-time) Markov chain. Standard Monte Carlo simulation techniques do not work well for rare events, so we use importance sampling; i.e., we change the probability measure governing the Markov chain such that transitions "towards" the goal state become more likely. To do this, we need an idea of distance to the goal state, so some level of knowledge of the Markov chain is required. In this article, we use graph analysis to obtain this knowledge. In particular, we focus on knowledge of the shortest paths (in terms of "rare" transitions) to the goal state. We show that only a subset of the (possibly huge) state space needs to be considered. This is effective when the high dependability of the system is primarily due to high component reliability, but less so when it is due to high redundancies. For several models, we compare our results to well-known importance sampling methods from the literature and demonstrate the large potential gains of our method.

Categories and Subject Descriptors: I.6 [**Computing Methodologies**]: Rare-event Simulation

General Terms: Algorithms, Theory

Additional Key Words and Phrases: Rare-event simulation, importance sampling, highly reliable systems

ACM Reference format:
Daniël Reijsbergen, Pieter-Tjerk De Boer, Werner Scheinhardt, and Sandeep Juneja. 2018. Path-ZVA: General, Efficient, and Automated Importance Sampling for Highly Reliable Markovian Systems. *ACM Trans. Model. Comput. Simul.* 28, 3, Article 22 (July 2018), 25 pages.
https://doi.org/10.1145/3161569

1 INTRODUCTION

Critical systems and infrastructures are increasingly required to be highly reliable, which has implications not only for the reliability of individual system components, but also for the accuracy of model-based evaluation. Realistic models of highly reliable systems typically have very large state spaces. Additionally, low component failure rates or a wide range of included system behaviours mean that a model may exhibit multiple *time scales*, in which system failure is the unlikely result of low-intensity state transitions (e.g., component failures) taking precedence over high-intensity

This work is partially supported by the Netherlands Organisation for Scientific Research (NWO), project number 612.064.812, and by the EU projects QUANTICOL, 600708, and SENSATION, 318490.
Authors' addresses: D. Reijsbergen, P.-T. De Boer, W. Scheinhardt, and S. Juneja; emails: daniel_reijsbergen@sutd.edu.sg, {p.t.deboer, w.r.w.scheinhardt}@utwente.nl, juneja@tifr.res.in.

https://doi.org/10.1145/3161569

transitions (e.g., component repairs). Numerical methods for evaluating system failure probabilities such as those implemented in the model checking tool PRISM (Kwiatkowska et al. 2011)—e.g., the Gauss-Seidel method—typically prove to be computationally infeasible due to the size of state space. Furthermore, state space reduction techniques that ignore low-intensity behaviour risk disposing of unlikely but interesting events.

A common and generally applicable alternative is *Monte Carlo simulation*, which only requires an implicit description of the state space and is therefore largely independent of its size. However, if the interesting behaviour is unlikely, a prohibitively large number of simulation runs is typically required before the rare event of interest is first observed. Hence, there is a need for hybrid techniques that strike a compromise between numerical techniques and standard Monte Carlo, whilst maintaining, to the largest possible extent, the general applicability of both methods.

In this article, we focus on Markovian systems in which individual components can fail and be repaired, and system failure occurs when certain combinations of components have failed. Crucially, we assume that the component failure rates have a (much) smaller order of magnitude than the repair rates. This is formalised in the notion of *highly reliable Markovian systems* (HRMSs), which include any Markov chain in which rates are parameterised by powers of some rarity parameter ϵ, where higher powers of ϵ correspond to component failures. Our (very small) probability of interest is that of reaching a system failure state within one regeneration cycle, i.e., between two visits to a given regeneration state. Once this quantity has been estimated, renewal theory (Cox 1962) can be used to calculate many system performance measures of practical interest, such as the mean time to failure, the unreliability, and the unavailability, without the need to estimate any other quantities that involve rare events.

Our starting point will be a discrete time Markov chain (DTMC) with fixed state space size and structure. When the HRMS is a Markov chain in continuous time (as is usually the case), we simply consider the DTMC embedded at transition times (replacing the transition rates by normalised transition probabilities). This is allowed since the probability of our interest does not depend on the times spent in states, and hence is the same in the embedded DTMC as in the original system.

To estimate rare event probabilities in the DTMC, we use *importance sampling*—a simulation method in which transitions that lead to the rare event are made more likely (Heidelberger 1995). More precisely, we follow a so-called Zero Variance Approximation (ZVA) scheme, based on some *a priori* approximation of the probability of interest. For this approximation we use *path-based* measures for the distance from each state to the target state, which is why we call our method "Path-ZVA." Our distance measure is the number of "failure" transitions needed to get to the target state, or, in general, the ϵ-power of the most likely path to get there, and during the simulation we will "push" the system in a direction that minimises this distance. It turns out that in many cases only a small part of the state space needs to be considered to find the *relevant* paths, making the method computationally advantageous. Hence, our method consists of *(i) a pre-processing step*, in which a graph-analysis algorithm finds the shortest paths on a subset of the state space, followed by *(ii) the actual simulations*, using an importance sampling scheme (based on the shortest paths) for efficiently estimating the probability of interest over the entire state space.

This Path-ZVA procedure is

(1) *general*, as the *only* requirements on the Markov model are that it is parameterised using ϵ-orders and that the relevant subset is numerically better tractable than the state space as a whole;
(2) *efficient*, as it provably has the desirable properties of either Bounded Relative Error and Bounded Normal Approximation, or Vanishing Relative Error for small ϵ; and
(3) *automated*, as the algorithm requires no user input apart from the model description. The code is available on http://datashare.is.ed.ac.uk/handle/10283/2630.

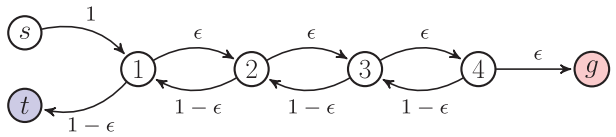

Fig. 1. Example of a HRMS with initial state s, goal state g, and regeneration state t.

The remainder of this article is as follows. After a formal description of the model setting and of (ZVA) importance sampling simulation in Section 2, we describe our "Path-ZVA algorithm" in detail in Section 3. Next, we prove in Section 4 that the resulting estimators have several desirable efficiency properties. We discuss a further variance reduction technique in Section 5, which makes optimal use of the pre-processing step. Finally, we present an empirical evaluation of all the discussed techniques in Section 6, including a comprehensive case study involving several benchmark models from the literature. Most of this article is based on Chapters 5 and 6 of Reijsbergen (2013). For a list of symbols used in this article, see Table 1.

2 MODEL AND PRELIMINARIES

2.1 Model Setting

The model is given in terms of a DTMC with a (large, possibly infinite) state space $\mathcal{X}$. (Note that the timing behaviour of the system is not important to the method; in fact, the DTMCs of the multicomponent systems of Section 6 are the underlying DTMCs of continuous-time Markov chains.) We assume that the system starts in a unique initial state $s \in \mathcal{X}$, and that there is a single goal state $g \in \mathcal{X}$ (potentially after merging all states from a bigger goal set into a single state). We also assume that there is (again after a potential merge) a single taboo/regeneration state $t \in \mathcal{X}$. Note that it may *not* be clear *a priori* which states are to be merged into g and t; since large DTMCs are typically described using a high-level language (e.g., a stochastic Petri net), we determine which states in the relevant part of the state space to collapse into g and t on-the-fly whilst running the algorithm described in Section 3. In the following, we assume that $s \neq g$ and $s \neq t$. We are no longer interested in the behaviour of the system once the system hits g or t, so we assume that these states are absorbing, i.e., have a self-loop with probability 1.

The complete transition probability structure in the DTMC is given by the probabilities p_{xz} of jumping from state x to state z, with $x, z \in \mathcal{X}$. The probabilities p_{xz} depend on ϵ, the time-scale parameter that formalises the notion that there are fundamental differences between the transition probabilities in the DTMC. To say more about the dependence on ϵ, we will write in the sequel, with $f, h, f_y, g_y : \mathbb{R} \to \mathbb{R}$:

$$
\begin{array}{lll}
f(\epsilon) = \Theta(h(\epsilon)) & \text{iff} & 0 < \lim_{\epsilon \downarrow 0} f(\epsilon)/h(\epsilon) < \infty, \\
f(\epsilon) = O(h(\epsilon)) & \text{iff} & \lim_{\epsilon \downarrow 0} f(\epsilon)/h(\epsilon) < \infty, \\
f(\epsilon) = o(h(\epsilon)) & \text{iff} & \lim_{\epsilon \downarrow 0} f(\epsilon)/h(\epsilon) = 0, \\
f_y(\epsilon) = \Theta(g_y(\epsilon)) \textit{ uniformly} \text{ in } y & \text{iff} & \exists a, b > 0 \text{ such that } \forall y : a < \lim_{\epsilon \downarrow 0} f_y(\epsilon)/g_y(\epsilon) < b,
\end{array}
$$

assuming these limits exist.

Throughout, our assumption is that for all non-zero transition probabilities $p_{xz} > 0$, some $r_{xz} \in \mathbb{N} \cup \{\infty\}$ exists such that $p_{xz}(\epsilon) = \Theta(\epsilon^{r_{xz}})$. If $p_{xz} = 0$, we set r_{xz} equal to ∞. Note that r_{xz} for $x, z \in \mathcal{X}$ are fixed parameters of the model. An example of a DTMC parameterised in this way can be found in Figure 1. (See Section 3.3 for a discussion on how these r_{xz} are chosen in practice.)

Let a *path* ω be a sequence $\omega(0), \omega(1), \ldots, \omega(n_\omega)$ of states in $\mathcal{X}$, with n_ω denoting the number of steps in the path. Let $\Omega(x)$ be the set of paths ω starting at $\omega(0) = x$. We are interested in the

Table 1. List of Symbols

$\mathcal{X}$	state space of the Markov chain
s, g, t	initial state, goal state, and taboo/regeneration state, respectively
$\mathbb{P}, p_{xz}$	original probability of the transition from state x to state z
$\mathbb{Q}, q_{xz}$	new probability of the transition from state x to state z
r_{xz}	ϵ-order of the transition from state x to state z, i.e., $p_{xz} = \Theta(\epsilon^{r_{xz}})$
ω	a path, i.e., a sequence of states $\omega(0), \omega(1), \ldots, \omega(n_\omega)$
$\mathbb{P}(\cdot)$	probability of a (set of) path(s) under p_{xz}
$\mathbb{Q}(\cdot)$	probability of a (set of) path(s) under q_{xz}
$\Omega(x)$	set of all paths ω starting at $\omega(0) = x$
$\Phi(x)$	set of all "successful" paths in $\Omega(x)$, in which g is reached before t
$\pi(x)$	$\sum_{\omega\in\Phi(x)} \mathbb{P}(\omega) = \mathbb{P}(\Phi(x))$
$d(x, z)$	shortest distance from x to z in terms of ϵ-orders
Λ	all states x for which $d(s, x) \leq d(s, g)$, including s and g
Γ	all states z in $\mathcal{X} \setminus \Lambda$ such that $\exists x \in \Lambda$ s.t. $p_{xz} > 0$
$\bar{p}_{xz}, \bar{\mathbb{P}}, \bar{d}$	$p_{xz}, \mathbb{P}, d$ as before, but in the system in which states in Γ have been given a transition with probability 1 to g
$\Delta(x)$	"dominant" paths $\omega \in \Phi(x)$, for which $\bar{\mathbb{P}}(\omega) = \Theta(\epsilon^{\bar{d}(x,g)})$
Φ, π, Δ	shorthand notation for $\Phi(s), \pi(s), \Delta(s)$
$v^\Delta(x)$	$\sum_{\omega\in\Delta(x)} \bar{\mathbb{P}}(\omega) = \bar{\mathbb{P}}(\Delta(x))$, approximation of $\pi(x)$

event that the system reaches g before t. To formalise this, let $\forall x \in \mathcal{X}$

$$\Phi(x) \triangleq \{\omega \in \Omega(x) : \omega(n_\omega) = g, \ \ \forall k < n_\omega : \omega(k) \notin \{t, g\}\}$$

be the set of all paths starting in x in which the event of interest occurs and which terminate as soon as g is reached. For all $x \in \mathcal{X}$, we define the probability that the rare event occurs, starting in x, as

$$\pi(x) \triangleq \sum_{\omega\in\Phi(x)} \mathbb{P}(\omega) = \mathbb{P}(\Phi(x)) \qquad \text{where} \qquad \mathbb{P}(\omega) \triangleq \prod_{i=1}^{n_\omega} p_{\omega(i-1)\omega(i)}. \tag{1}$$

We are interested in $\pi \triangleq \pi(s)$ and estimate this probability using simulation, as described below.

2.2 Simulation

The basic means to evaluate π is through a point estimate; to obtain one, we draw $N \in \mathbb{N}$ sample paths to obtain a sample set $\{\omega_1, \ldots, \omega_N\}$. To draw a sample path, we start in s and draw successor states using $\mathbb{P}$ until we reach either t or g. (We assume that this happens in finite time with probability 1.) Let $\Phi \triangleq \Phi(s)$, and $\mathbf{1}_\Phi(\omega)$ denote an indicator function which equals 1 if ω is in Φ and 0 otherwise; this allows us to obtain an unbiased estimator of π, given by

$$\hat{\pi}_\mathbb{P} = \frac{1}{N} \sum_{k=1}^{N} \mathbf{1}_\Phi(\omega_k). \tag{2}$$

An approximate 95%-confidence interval for π can be obtained using the Central Limit Theorem (see, e.g., Law and Kelton (1991, §4.5))

As we discussed in the Introduction, we use importance sampling: we simulate using different transition probabilities $(q_{xz})_{x,z\in\mathcal{X}}$ under which paths in Φ are more likely. Let $\mathbb{Q}$ be the probability measure on paths defined analogously to $\mathbb{P}$ but for q_{xz}. We compensate for overestimation by

weighting each outcome with the ratio of $\mathbb{P}$ and $\mathbb{Q}$. Every time a transition is sampled using the new probabilities, this weighting factor needs to be incorporated. Our new estimator—replacing Equation (2)—then becomes

$$\hat{\pi}_{\mathbb{Q}} = \frac{1}{N}\sum_{k=1}^{N} L_{\mathbb{Q}}(\omega_k)\cdot \mathbf{1}_{\Phi}(\omega_k), \quad \text{with} \quad L_{\mathbb{Q}}(\omega) = \prod_{i=1}^{n_\omega} \frac{p_{\omega(i-1)\omega(i)}}{q_{\omega(i-1)\omega(i)}}. \tag{3}$$

This estimator is unbiased for any new distribution that assigns positive probability to transitions that have positive probability under the old distribution on paths in $\Phi(s)$ (by the Radon-Nikodym Theorem, see Chapter 7 of Capiński and Kopp (2004)). In the following, we will write $\hat{\pi} = \hat{\pi}_{\mathbb{Q}}$ for brevity.

If $\mathbb{Q}$ is chosen carefully, the estimator based on Equation (3) will have a lower variance than the standard estimator. The performance of an importance sampling method is measured by the variance of $\hat{\pi}$ under $\mathbb{Q}$, given by

$$\mathrm{Var}_{\mathbb{Q}}(\hat{\pi}) = \mathbb{E}_{\mathbb{Q}}\left(L_{\mathbb{Q}}^2 \cdot \mathbf{1}_{\Phi}\right) - \pi^2.$$

Using $\mathbb{Q} = \mathbb{P}$, we obtain the variance of the standard estimator: $\pi(1-\pi)$. A particularly interesting efficiency metric for an estimator is its *relative error*, given by

$$\frac{\sqrt{\mathrm{Var}_{\mathbb{Q}}(\hat{\pi})}}{\pi}.$$

The relative error of the standard estimator is given by $\sqrt{(1-\pi)/\pi}$, which goes to infinity when π goes to zero. When the relative error of an estimator remains bounded when π goes to zero, we say that our estimator has the desirable property of *Bounded Relative Error* (BRE). When it goes to zero, we say that it has the even more desirable property of *Vanishing Relative Error* (VRE) (L'Ecuyer et al. 2010).

We use the ZVA approach (cf. L'Ecuyer and Tuffin (2008)), and present the following probability measure $\mathbb{Q}$:

$$q_{xz} \triangleq \frac{p_{xz}\upsilon(z)}{\sum_{x' \in \mathcal{X}} p_{xx'}\upsilon(x')}, \tag{4}$$

where $\upsilon(z)$ is some *approximation* for the true probability $\pi(z)$. Clearly, if $\upsilon(z)$ were exactly equal to $\pi(z)$, the denominator would be $\pi(x)$ and the estimator would have zero variance (de Boer et al. 2007), but of course we do not explicitly know $\pi(\cdot)$. If the simulation distribution $\mathbb{Q}$ associated with the approximation υ is good enough, then we have succeeded in overcoming the main problem facing standard Monte Carlo simulation of rare events. The particular ZVA technique (choice of υ) discussed in this article—namely, Path-ZVA—will be the subject of Section 3.

2.3 Related Work

In this section, we give a brief overview of papers on the use of importance sampling for Highly Reliable Markovian Systems that we consider to be particularly relevant to this article, either because they discuss literature benchmarks or because they discuss recent advances. As a first remark, note that our notion of an HRMS (namely, any Markov chain in which the transitions are given ϵ-orders) is more general than what is typically considered in the literature. In the literature, an HRMS is often restricted to what we call a *multicomponent system*, where only failure transitions have rates of order ϵ, while we do not have this restriction.

The first application of an importance sampling method—namely, *failure biasing*—to HRMSs goes back to Lewis and Böhm (1984). The general notion of failure biasing means that greater probability is assigned to "*failures*," i.e., transitions that are chosen with a probability that is $O(\epsilon)$. Shahabuddin (1994) studied the asymptotic properties of a refined version of failure biasing called

balanced failure biasing (BFB), and showed the method to satisfy BRE in the absence of so-called High Probability Cycles (HPCs). Nakayama (1996) derived general conditions for BRE in importance sampling schemes for HRMSs. Carrasco (1992) proposed a method called *failure distance biasing*, in which the simulation measure is based on the distance from each state to the rare states. This distance notion is similar to the function d discussed in Section 3.1—given d, the method applies a form of failure biasing (with the exception that if a failure does not lead to a decrease in d, it is not treated as a failure). The function d in their setting is computed by finding the minimal cuts in the model's corresponding fault tree, which means the setting is limited (namely, multicomponent systems with independent component types, and no HPCs). Carrasco (2006) extended this approach to "unbalanced" systems. Alexopoulos and Shultes (2001) proposed a method that is based on bounding the value of the likelihood ratios, and which has good performance for both highly reliable and highly redundant systems. Juneja and Shahabuddin (2001) proposed a scheme—the *implementable general biasing scheme* (IGBS)—to mitigate the effects of HPCs on the performance of BFB.

We will use BFB and IGBS as literature benchmarks for the experiments of Section 6, so we discuss them in more detail in the following. In particular, for each state $x \in \mathcal{X}$, let $n_f(x)$ be the number of transitions leaving x with a positive ϵ-order (the "*failures*") and let $n_r(x)$ be the number of transitions leaving x with ϵ-order 0 (the "*repairs*"). Given some $p > 0$, the simulation measure $\mathbb{Q}$ of BFB is given by

$$q_{xz} = \begin{cases} (n_f(x))^{-1} & \text{if } n_r(x) = 0, \\ (n_r(x))^{-1} & \text{if } n_f(x) = 0, \\ p(n_f(x))^{-1} & \text{if } r_{xz} > 0 \text{ and } n_r(x) > 0, \\ (1-p)(n_r(x))^{-1} & \text{if } r_{xz} = 0 \text{ and } n_f(x) > 0. \end{cases}$$

The typical choice for p is $\frac{1}{2}$, and we make the same choice in this article. IGBS is similar to BFB, with the exception that the degree of biasing is reduced when the current state is part of an HPC. To avoid having to run a numerical procedure to detect HPCs, IGBS switches to low-intensity biasing when the previous transition was a high-probability transition (resulting in a non-Markovian simulation measure). In particular, with $q_{xz|x'} = \mathbb{Q}(\omega(i+1) = z \mid \omega(i) = x,\ \omega(i-1) = x')$, IGBS means

$$q_{xz|x'} = \begin{cases} (n_f(x))^{-1} & \text{if } n_r(x) = 0, \\ (n_r(x))^{-1} & \text{if } n_f(x) = 0, \\ p(n_f(x))^{-1} & \text{if } r_{xz} > 0, r_{x'x} > 0 \text{ and } n_r(x) > 0, \\ (1-p)(n_r(x))^{-1} & \text{if } r_{xz} = 0, r_{x'x} > 0 \text{ and } n_f(x) > 0, \\ \delta(n_f(x))^{-1} & \text{if } r_{xz} > 0, r_{x'x} = 0 \text{ and } n_r(x) > 0, \\ (1-\delta)(n_r(x))^{-1} & \text{if } r_{xz} = 0, r_{x'x} = 0 \text{ and } n_f(x) > 0, \end{cases}$$

for some $\delta < p$. In the initial state, p is used as a biasing constant. We choose $\delta = \frac{1}{100}$ in this article. Note that the measure described above is more general than Shahabuddin (1994), who assumed that $\forall x \in \mathcal{X} \setminus \{s, g, t\}$, $n_f(x) > 0$, and $n_r(x) > 0$.

In addition to the papers on ZVA mentioned in Section 2.2, L'Ecuyer and Tuffin (2011) discusses the particular application of ZVA to HRMSs. We use several of the ideas therein in Section 4. In particular, conditions are derived for a change of measure to satisfy VRE. In said paper, the analogues of d and v were not obtained explicitly, but approximated using the structure of multicomponent systems.

The basic idea underlying Section 5 is from Juneja (2007), who showed that for geometric sums of heavy-tailed random variables, a separation of the estimator into the numerical computation of a dominant component and the simulation of the small component yields an estimator with VRE.

Other contributions involving generally applicable efficient simulation of HRMSs include Budde et al. (2015), in which the notion of distance to the goal set used is the smallest possible number of transitions needed (which is equivalent to the model setting of this article if all transitions have ϵ-order 1). Another generic importance sampling technique is the *cross-entropy method* (see, e.g., Ridder (2010)), which we do not discuss further in this article because of its heuristic nature.

3 THE PATH-ZVA ALGORITHM

In this section, we describe the simulation method of this article—Path-ZVA. We discuss two versions, ZVA-$\bar{d}$ and ZVA-Δ, which differ in the distance measure used. In the following, we first give a formal description of these two methods and the underlying concepts. We then discuss their implementation, with a particular focus on the routines of Algorithms 1, 2, and 3.

3.1 Path-Based ZVA

Our method for finding a suitable approximation v of π is to select only a subset of the paths used in the summation of Equation (1), namely, the so-called *dominant* paths, as we discuss below. In order to determine which paths to select, we will define two related measures—$\bar{d}$ and v^{Δ}—for the distance between each state and the rare state g. Throughout this subsection, we assume that no so-called HPC is present, where we define a HPC (see also Section 2.3) as a cyclic path ω with $\omega(n_\omega) = \omega(0)$ and $\mathbb{P}(\omega) = \Theta(\epsilon^0)$. For Markov chains that do have one or more HPCs, we explain in Section 3.2 how these are removed.

First, we define the function $d : \mathcal{X}^2 \to \mathbb{N}$ as

$$d(x,z) = \min\{r : \exists \omega \in \Omega(x) \text{ with } \omega(n_\omega) = z,\ \ \forall k < n_\omega : \omega(k) \notin \{t, g\} \ \text{ and } \ \mathbb{P}(\omega) = \Theta(\epsilon^r)\}.$$

Intuitively, $d(x,z)$ is the shortest ϵ-distance of any path from x to z. Of particular interest are $d(x,g)$, the shortest distance from each state $x \in \mathcal{X}$ to the goal state, and $d(s,x)$, the shortest distance from the initial state to each state x.

As mentioned in the Introduction, we do not need to run the algorithm on the entire state space, but only the states that are asymptotically at most as hard to reach from s as g, and their neighbours. To formalise this, we introduce the following two sets:

$$\Lambda = \{x \in \mathcal{X} : d(s,x) \le d(s,g)\} \quad \text{and} \tag{5}$$

$$\Gamma = \{x \in \mathcal{X} \setminus \Lambda : \exists z \in \Lambda \text{ s.t. } p_{zx} > 0\}.$$

In words: Λ is the relevant part of $\mathcal{X}$, i.e., the set of states that are asymptotically not substantially less likely to be reached from s than g. The set Γ contains the states "bordering" Λ, i.e., those states to which the system can jump directly from Λ. By construction, $d(s,x) > d(s,g)$ for all $x \in \Gamma$. We assume that both Λ and Γ are finite—if they are not, the numerical pre-processing phase will never terminate. See Figure 2 for an illustration of the sets Lambda and Gamma.

The algorithm of this article calculates $d(s,x)$, $d(x,g)$, and $v(x)$ only for $x \in \Lambda$. This means that Equation (4) cannot be applied when $x \in \Lambda$ and $z \in \Gamma$. This is remedied by adapting $\mathbb{P}$ to an alternative probability measure $\bar{\mathbb{P}}$ with high-probability "shortcuts" from Γ to g, and its corresponding distance measure $\bar{d}$. First, let $\bar{p}_{xz}$ be defined as follows:

$$\bar{p}_{xz} = \begin{cases} p_{xz} & \text{if } x \notin \Gamma, \\ 1 & \text{if } x \in \Gamma \text{ and } z = g, \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

Then we let $\bar{\mathbb{P}}$ and $\bar{d}$ be defined as $\mathbb{P}$ and d under this new measure. Next, we define

$$\Delta(x) = \{\omega \in \Phi(x) : \bar{\mathbb{P}}(\omega) = \Theta(\epsilon^{\bar{d}(x,g)})\},$$

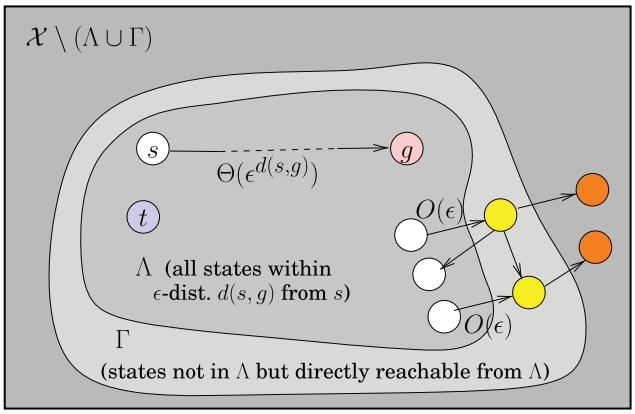

Fig. 2. Illustration of the sets Λ and Γ within $\mathcal{X}$.

the set of paths from x to the goal state g that have (under $\bar{\mathbb{P}}$) the minimal distance $\bar{d}(x,g)$. As before, we compute $\bar{d}(s,x)$, $\bar{d}(x,g)$, and $\Delta(x)$ only for state x if $x \in \Lambda$. (Note that, even though we allow $\Omega(x)$ and $\Phi(x)$ to include "paths" that have probability zero under $\mathbb{P}$, such paths are never in $\Delta(x)$, since either they include one or more transitions with probability zero under $\bar{\mathbb{P}}$, or they traverse Γ and their ϵ-order exceeds $\bar{d}(x,g)$.) We call the paths in $\Delta(x)$ the *dominant* paths from x to the goal state. Finally, we define the function $v^{\Delta} : \mathcal{X} \to \mathbb{R}^+$ as the probability of the "dominant" paths under $\bar{\mathbb{P}}$:

$$v^{\Delta}(x) = \sum_{\omega \in \Delta(x)} \bar{\mathbb{P}}(\omega). \tag{7}$$

The function v^{Δ} can be substituted for v in Equation (4) to yield a well-performing simulation measure. This approach will be called ZVA-Δ in this article. Alternatively, one can use $v(x) = \epsilon^{\bar{d}(x,g)}$, which is easier to compute and, as we will see in Section 4, still yields an estimator with favourable properties. This approach will be called ZVA-$\bar{d}$ in this article. Note that there are model settings for which techniques exist that allow for ZVA-$\bar{d}$ to be applied without the need to consider each individual state in Λ: see, e.g., Reijsbergen et al. (2013) for an application to stochastic Petri nets. In the approach of that paper, the full state space is partitioned into "zones" such that for each zone it holds that $\bar{d}$ in each state is given by the same affine function of the state vector. The performances of ZVA-Δ and ZVA-$\bar{d}$ will be compared in Section 6.

Regardless of the choice of v, when we leave Λ during the simulation we *stop using importance sampling* and revert back to standard Monte Carlo until we reach either g or t. A consequence is that the simulation measure $\mathbb{Q}$ is now *non-Markovian*: it is only Markovian as long as we stay in Λ. Let

$$m(\omega) = \min\{i \in \mathbb{N} : \omega(i) \notin \Lambda \text{ or } \omega(i) = g\}. \tag{8}$$

Then $\mathbb{Q}$ is as follows (replacing Equation (4)):

$$q_{\omega(i)\omega(i+1)} = \begin{cases} \dfrac{\bar{p}_{\omega(i)\omega(i+1)} v(\omega(i+1))}{\sum_{z \in \mathcal{X}} \bar{p}_{\omega(i)z} v(z)} & \text{if } i < m(\omega) \\ p_{\omega(i)\omega(i+1)} & \text{if } i \geq m(\omega). \end{cases} \tag{9}$$

3.2 Pre-Processing: Graph Analysis Procedure for Finding $\bar{d}$ and v^{Λ}

The algorithm for determining $\bar{d}$ and v^{Λ} involves the search for a shortest path in a graph, and is strongly inspired by Dijkstra's method (Dijkstra 1959). The new algorithm can be broken down into three main routines, namely, Algorithms 1, 2, and 3. Unlike Dijkstra's algorithm, the algorithm of this section consists of two phases: a forward phase and a backward phase. In the forward phase, we generate the state space and remove HPCs until we have found g and Λ, and in the backward phase we start in g and determine $\bar{d}$ and v^{Λ} by working back until we reach s. The forward phase is described in Algorithm 1 and the backward phase is described in Algorithm 3. Algorithm 2 removes a detected HPC and is called by Algorithm 1. The runtimes of all the algorithms are polynomial in the size of $\Lambda \cup \Gamma$.

3.2.1 Forward Phase. In the first phase, we use a procedure based on Dijkstra's algorithm for finding shortest paths in a graph in order to determine $\bar{d}(s, \cdot)$, Λ, and to remove all HPCs. In particular, $\bar{d}(s, \cdot)$ is used to detect the HPCs; it is denoted by $\bar{d}'(\cdot)$ in Algorithm 1 for brevity.

ALGORITHM 1: Forward phase.

Require: Markov chain $(\mathcal{X}, P)$ with $P = (p_{xz})_{x,z\in\mathcal{X}}$, source s, destination g.

1: $\Lambda := \emptyset$
2: $\bar{d}'(s) := 0, \quad \forall z \in \mathcal{X} \setminus \{s\} : \bar{d}'(z) := \infty$
3: $P' := P, \quad x := s$
4: **while** $\bar{d}'(x) \leq \bar{d}'(g)$ **do**
5: $\quad \Lambda := \Lambda \cup \{x\}$
6: $\quad$ **for all** $z \in \mathcal{X}$ s.t. $p_{xz} > 0$ **do**
7: $\quad\quad \bar{d}'(z) := \min(\bar{d}'(z), \bar{d}'(x) + r_{xz})$
8: $\quad\quad$ **if** $z \in \Lambda$ and $\bar{d}'(z) = \bar{d}'(x)$ **then**
9: $\quad\quad\quad P' := \text{loopDetect}((\mathcal{X}, P'), z)$
10: $\quad\quad$ **end if**
11: $\quad$ **end for**
12: $\quad x := \arg\min\{\bar{d}'(z) : z \in \mathcal{X} \setminus \Lambda\}$ ▷ if several states are possible
13: **end while** ▷ in line 12, any can be chosen
14: **return** Λ, P'

Whilst running the procedure, we iteratively update Λ—this allows us to use Λ to keep track of the visited states. We initialise $\Lambda = \emptyset$ and $\bar{d}'(s) = 0$. We set the current state x equal to s. Then, we carry out the following routine until x equals g: we add x to Λ, and set $\bar{d}'(z) = \min(\bar{d}'(z), \bar{d}'(x) + r_{xz})$ for each possible successor state z of x— i.e., we let the new best value for $\bar{d}'(z)$ be the minimum between the old best value and the new possible value. We then set x equal to the state z that has not been considered before with the lowest value of $\bar{d}'$, and start over. When we have reached g, we complete the procedure for all states z with $\bar{d}'(z) = \bar{d}'(g)$ before we terminate the first phase. The set Λ then meets its definition given in Equation (5).

If, whilst running the procedure, we find that a state z has a successor state z' such that $\bar{d}'(z) = \bar{d}'(z')$, we trigger the loop-detection procedure of Algorithm 2. It essentially boils down to removing all low-probability transitions from the relevant part of the DTMC and finding the Strongly Connected Component (SCC) that contains the states z and z' that triggered the procedure, using the algorithm from Barnat et al. (2011). Essentially, we determine A, the set of states that can be reached from z using high-probability transitions, and B, the set of states from which z can be reached using high-probability transitions. The relevant SCC is then $A \cap B$.

In Algorithm 2, we find A through the set S_A which contains those states added to A in each step. After initialising $S_A = \{x\}$, we iteratively find those states that can be reached from the states

ALGORITHM 2: loopDetect$((\mathcal{X}, P), x')$.

Require: Markov chain $(\mathcal{X}, P)$, state x'.

1: $P' := P$
2: $A := \emptyset, B := \emptyset$
3: $S_A := \{x'\}, S_B := \{x'\}$
4: **while** $S_A \neq \emptyset$ and $S_B \neq \emptyset$ **do**
5: $\quad A := A \cup S_A, B := B \cup S_B$
6: $\quad S'_A := S_A, S_A := \{z \in \mathcal{X} \setminus A \ : \ \exists x \in S'_A \text{ s.t. } r_{xz} = 0\}$
7: $\quad S'_B := S_B, S_B := \{z \in \mathcal{X} \setminus B \ : \ \exists x \in S'_B \text{ s.t. } r_{zx} = 0\}$
8: **end while**
9: **while** $S_A \neq \emptyset$ **do**
10: $\quad A := A \cup S_A$
11: $\quad S'_A := S_A, S_A := \{z \in B \setminus A \ : \ \exists x \in S'_A \text{ s.t. } r_{xz} = 0\}$
12: **end while**
13: **while** $S_B \neq \emptyset$ **do**
14: $\quad B := B \cup S_B$
15: $\quad S'_B := S_B, S_B := \{z \in A \setminus B \ : \ \exists x \in S'_B \text{ s.t. } r_{zx} = 0\}$
16: **end while**
17: $L := A \cap B$
18: $D := \{x \in \mathcal{X} \setminus L : \exists z \in L \text{ s.t. } p'_{zx} > 0\}$
19: Solve $\begin{bmatrix} \mu_{xz} = p_{xz} + \sum_{z' \in L} p_{xz'} \mu_{z'z}, & \forall x \in L, z \in D, \\ 1 = \sum_{z' \in D} \mu_{xz'}, & \forall x \in L \end{bmatrix}$ for μ_{xz}
20: $p'_{xz} := \mu_{xz}, \forall x \in L, z \in D, \quad p'_{xz} := 0, \forall x \in L, z \in L$
21: **return** P'.

in the previous iteration of S_A (denoted by S'_A in the algorithm) using high-probability transitions. We terminate when no more states can be added, i.e., when S_A equals $\emptyset$. This is done in lines 9–12; we do something similar for B, S_B, and S'_B in lines 13–16. These lines are preceded by lines 4–8 in which we combine A and B. The reason behind this combined phase is that B is potentially (much) larger than Λ and Γ; it may even be infinite. In order to avoid the algorithm's non-termination due to this complication, we alternate between carrying out a step for A and a step for B in lines 4–8. If we can no longer find new candidates for A, then A has been determined. Since states in the HPC need to be both in A and B, we from then on only select candidates for B that are in A. We terminate if we can no longer find candidates for B in A. The same is done for A and B interchanged. This way, we always terminate in a finite amount of time because $A \subset \Lambda$ and Λ is finite.

Having determined the SCC, we construct a new DTMC with the same state space and identical rare event probabilities $\pi(x)\ \forall x \in \mathcal{X}$, but with the transition probabilities of the states in the HPC redistributed. This can be done using a SCC-based state space reduction technique similar to the one described by Ábrahám et al. (2010), implemented in line 19 of Algorithm 2. In our implementation, the system of equations in line 19 is approximately solved using Gauss-Seidel. Algorithm 2 is repeated each time a new HPC is detected.

3.2.2 Backward Phase. In this phase, we determine v^{Λ} and $\bar{d}(\cdot, g)$; the latter is denoted by $\bar{d}^*(\cdot)$ in Algorithm 3. We initiate the second phase in g (since g is given implicitly through a high-level description, this would not have been possible without the first phase). We use a list Λ' to keep track of the states that have been considered, and initialise Λ', v^{Λ}, and $\bar{d}$ as outlined in the beginning of Algorithm 3. For each predecessor x of g that is in $\Lambda \cup \Gamma$, we add x to Λ' if this had not been done already and if $\bar{d}(x) = r_{xg}$, we update $v^{\Lambda}(x) := v^{\Lambda}(x) + p'_{xg}$. We then choose the next state to consider: this is the state x in $(\Lambda \cup \Gamma) \setminus \Lambda'$ (i.e., the set of states that have not yet been considered)

ALGORITHM 3: Backward phase.

Require: Markov chain (Λ, P'), end node g.
1: $\forall z \in \Lambda: \ v^{\Lambda}(z) := 0, \bar{d}^*(z) := \infty$
2: $v^{\Lambda}(g) := 1, \bar{d}^*(g) := 0$
3: $\Lambda' := \emptyset, x := g$
4: $\Gamma := \{x \in \mathcal{X} \setminus \Lambda : \exists z \in \Lambda \text{ s.t. } p_{zx} > 0\}$
5: **while** $\Lambda' \neq \Lambda \cup \Gamma$ **do**
6: $\quad x := \arg\min\{\bar{d}^*(x) : x \in (\Lambda \cup \Gamma) \setminus \Lambda' \text{ and } \nexists x' \in (\Lambda \cup \Gamma) \setminus \Lambda' \text{ s.t. } r_{x'x} = 0\}$
7: $\quad$ **for all** $z \in \Lambda \cup \Gamma$ **do** ▷ if several states are possible
8: $\quad\quad$ **if** $r_{zx} + \bar{d}^*(x) < \bar{d}^*(z)$ **then** $v^{\Lambda}(z) := 0$ ▷ in line 6, any can be chosen.
9: $\quad\quad \bar{d}^*(z) := \min(\bar{d}^*(z), r_{zx} + \bar{d}^*(x))$
10: $\quad\quad$ **if** $\bar{d}^*(z) = \bar{d}^*(x) + r_{zx}$ **then**
11: $\quad\quad\quad v^{\Lambda}(z) := v^{\Lambda}(z) + p'_{zx} v^{\Lambda}(x)$
12: $\quad\quad$ **end if**
13: $\quad$ **end for**
14: $\quad \Lambda' := \Lambda' \cup \{x\}$
15: **end while**
16: **return** $\bar{d}^*, v^{\Lambda}, \Gamma$.

for which $\bar{d}$ is the lowest *and* for which no other state z in $(\Lambda \cup \Gamma) \setminus \Lambda'$ exists for which $r_{xz} = 0$. The reason is that otherwise, the probability of the paths going from x to z is never added to $v^{\Lambda}(x)$, which has a cascading effect on the predecessors of x. Note that we can always find such a state only if the HPCs have been removed. We continue performing the same procedure until we have determined $v^{\Lambda}(x)$ for all $x \in \Lambda \cup \Gamma$.

3.3 Practical Aspects of the Path-ZVA Algorithm

Identifying ϵ in Practical Models. In principle, the algorithms described above can be applied to any DTMC with transition probabilities that are parameterised by powers of some small parameter ϵ. Usage of ϵ-powers for the purpose of analysing the efficiency of simulation algorithms goes back to at least Shahabuddin (1994). However, in our case (and earlier; see de Boer et al. (2007)) the change of measure *itself* depends on the ϵ-powers. This means that a practitioner who has a model with given rates/probabilities will need to assign ϵ-powers to them, which can be done in infinitely many ways.

There are a few trivial approaches that do not work well, but are illustrative. One is to simply set the ϵ-power to 0 for all transitions, and represent the model entirely by the pre-factors $\lambda_{xz} = p_{xz}/\epsilon^{r_{xz}}$. Then our algorithm will treat the model as one large HPC, and the probability of interest will be computed numerically if the state space is sufficiently small. The other extreme is to set all pre-factors to 1, choose a value of ϵ just below 1, and represent the model entirely by (very high) exponents r_{xz}. Then the algorithm will focus the simulation effort on the single most likely path, at the expense of paths which are only slightly (namely, by a factor of ϵ) less likely, causing underestimation and/or high variance. A third approach is to set all ϵ-powers to 1, as is done by Budde et al. (2015). Although this is a more natural approach than the other two, it still does not distinguish between failures and repairs.

In typical reliability models, repair rates are several orders of magnitude higher than failure rates. In such cases, giving component repairs ϵ-order 0 and failures ϵ-order 1 is typically a good choice. If some failures are very much less likely than others (this is a feature of so-called "unbalanced" systems), higher ϵ-orders can be assigned to those to achieve further variance reduction

Table 2. Total and Reduced State Space Sizes and the Pre-Processing Sets Λ for a Range of Models

Model	Source	$\lvert\mathcal{X}\rvert$ (total)	$\lvert\mathcal{U}\rvert$	$\lvert\Lambda\rvert$	$\lvert\Gamma\rvert$
2-node tandem queue, overflow level n	(R)	∞	∞	$O(n^2)$	$O(n)$
Distrib. Datab. Syst. (dedicated repair)	many; (S)	421,875	514	48	84
Distributed Database System (FCFS)	see (S)	2,123,047,371	>500,000	84	504
k-out-of-n system (homogeneous)	(A)	$O(n)$	$O(k)$	$O(k)$	0
k-out-of-n system (heterogeneous)		$O(2^n)$	$O(2^k)$	$O(2^k)$	0
Fault-Tolerant Database System	(C)	14,762,250,000	59,051	87	1,060
Fault-Tolerant Control System	(C)	1,855,425,871,872	>500,000	116	2,928
Network with Redundancies	(A)	very large	very large	still very large	

More information can be found in the following sources: (R) Reijsbergen et al. (2013), (S) Section 6.2.1, (A) Alexopoulos and Shultes (2001), and (C) Carrasco (2006). The >500,000 entries for $\lvert\mathcal{U}\rvert$ are lower bounds established by 12 hours of computation.

(see Shahabuddin (1994, Fig. 1)). This approach can be automated to a large extent by having the practitioner specify only ϵ beforehand, and assigning the smallest integer ϵ-power to each transition such that its pre-factor is greater than ϵ. This is, in fact, what we have implemented and applied in Section 6.2. Carrasco (2006) chooses ϵ as the ratio of the largest failure rate to the smallest repair rate. Further experimentation to establish best practice with regard to choosing ϵ is an interesting direction for further research.

Numerical Complexity. The numerical complexity of the phases of our algorithm is as follows. Let D be the maximum number of successors of all states in Λ (this is $|\Lambda \cup \Gamma|$ at worst but usually much smaller). The loop in line 4 of Algorithm 1 has $|\Lambda|$ iterations, and the nested loop in line 6 has D iterations, so the total complexity is $O(D|\Lambda|)$. Lines 4–16 of Algorithm 2 have complexity $O(\max|L|)$, where $\max|L|$ denotes the size of the largest HPC plus direct predecessors and successors. Line 19 of Algorithm 2 has a complexity of $O((\max|L|)^2)$ if implemented using the approximative Gauss-Seidel algorithm. Line 5 of Algorithm 3 has $|\Lambda \cup \Gamma|$ iterations, and although the nested loop in line 7 only has to be done for the number of predecessors in each state, these two loops together will have total complexity $O(D|\Lambda \cup \Gamma|)$ since the total number of incoming and outgoing transitions within $\Lambda \cup \Gamma$ is the same.

In summary, the complexity of our algorithm is typically $O(D|\Lambda \cup \Gamma|)$ or $O(|\Lambda \cup \Gamma|^2)$. This is to be compared to the cost of computing the probability of interest without simulation, which is typically $O(D|\mathcal{U}|)$ or $O(|\mathcal{U}|^2)$, where $\mathcal{U}$ is what remains of the full state space $\mathcal{X}$ after collapsing all goal states (and states that can only be reached via goal states) into a single state g. Hence, what we gain is that we apply numerical analysis only to $\Lambda \cup \Gamma$ rather than to $\mathcal{U}$. This is illustrated in Table 2 for a range of models.

High Component Reliability versus High Redundancy. For models whose high reliability is mostly due to high redundancy, the method tends to be less effective. One reason is that $|\Lambda|$ is large in such models; this is apparent in the last line, and potentially also the fifth line (depending on the value of k), of Table 2 . The other reason is that when many "almost-dominant" paths exist, of order $\epsilon^{\tilde{d}(s,g)+1}$ or higher, their total contribution may dominate the (fewer) supposedly "dominant" path(s) of order $\epsilon^{d(s,g)}$, if ϵ is not small enough. This can easily happen in models of highly redundant systems, with, e.g., many different possible sequences of failure and repair events on those almost-dominant paths, and ϵ tending to be larger because of larger individual component failure rates.

Efficient Implementation. A crude way of implementing the method would involve constructing the entire state space and keeping track of matrices giving the transition probabilities and ϵ powers for each combination of states. However, this would be very memory-inefficient, or impossible in the case of an infinite state space. Specification of a model in our implementation consists only of three functions that determine, given a state, (1) whether it is a goal state, (2) whether it is a taboo state, and (3) three arrays specifying its successors' state indices, the probabilities of jumping to these successors (typically implicitly through CTMC rates), and the corresponding ϵ-powers. Our implementation also allows for the last array to be omitted and the ϵ-orders to be computed using a given value ϵ in the manner discussed previously. There is no need to generate the entire state space; states only need to be considered "on-the-fly," as they are encountered during pre-processing and the actual simulation.

4 ASYMPTOTIC PERFORMANCE OF THE ESTIMATOR

In this section, we consider the performance of the two versions of the estimator produced by the algorithm of Section 3. If the estimator is based on v^{Λ}, we show it has VRE (Theorem 4.6); if it is based on $\bar{d}$ (which is easier to compute), it does not necessarily have VRE, but it does have both BRE (Theorem 4.5) and the "Bounded Normal Approximation" property (Theorem 4.7) We first prove the technical Lemmas 4.1–4.4 before proving the main theorems.

Lemma 4.1. *If $v(x) = \Theta(\epsilon^{\bar{d}(x,g)})$ uniformly in x, then for all $x \in \Lambda$ we have that*

$$\sum_{z \in \mathcal{X}} \bar{p}_{xz} v(z) = \sum_{z \in \Lambda \cup \Gamma} \bar{p}_{xz} \Theta(\epsilon^{\bar{d}(z,g)}) = \Theta(\epsilon^{\bar{d}(x,g)})$$

uniformly in x.

Proof. Since $\bar{p}_{xz} = 0$ for $z \notin \Lambda \cup \Gamma$ and since $\Lambda \cup \Gamma$ is finite, the ϵ-order of the sum equals the ϵ-order of its largest element. Let z' be a state such that $\bar{p}_{xz'}\Theta(\epsilon^{\bar{d}(z',g)})$ has the lowest ϵ-order in the sum. Suppose that its ϵ-order is smaller than $\bar{d}(x,g)$, then there exists a path from x via z' to g with cost lower than $\bar{d}(x,g)$, which contradicts the definition of $\bar{d}(x,g)$. The uniformity follows trivially from the finiteness of $\Lambda \cup \Gamma$. □

Lemma 4.2.

$$d(s,g) = \bar{d}(s,g).$$

Proof. By the definition of Λ, any state $x \notin \Lambda$ has $d(s,x) > d(s,g)$ and $\bar{d}(s,x) > d(s,g)$, so any path leaving Λ has length $> d(s,g)$, both under $\mathbb{P}$ and $\bar{\mathbb{P}}$. Therefore, the shortest path from s to g under $\mathbb{P}$ must lie entirely inside Λ, and its length under $\bar{\mathbb{P}}$ is $d(s,g)$ too. Finally, any other path from s to g under $\bar{\mathbb{P}}$ cannot be shorter than $d(s,g)$: if it does not leave Λ, its length is the same under $\bar{\mathbb{P}}$ and $\mathbb{P}$, while if it leaves Λ, its length exceeds $d(s,g)$. □

Lemma 4.3. *If $v(x) = \Theta(\epsilon^{\bar{d}(x,g)})$ uniformly in x, then for any path ω starting in s and ending in g or Γ before leaving $\Lambda \cup \Gamma$, we have*

$$\frac{\mathbb{P}(\omega)}{\mathbb{Q}(\omega)} = \Theta(\epsilon^{d(s,g)}) \quad \textit{and, more specifically,} \quad \frac{\mathbb{P}(\omega)}{\mathbb{Q}(\omega)} \leq c_0^r \epsilon^{d(s,g)}$$

for some positive c_0, independent of ω, and with r the epsilon-order of ω.

Proof. Observe that

$$\frac{\mathbb{P}(\omega)}{\mathbb{Q}(\omega)} = \prod_{i=1}^{n_\omega} \frac{\bar{p}_{\omega(i-1)\omega(i)}}{q_{\omega(i-1)\omega(i)}} = \prod_{i=1}^{n_\omega} \frac{\sum_{z\in\mathcal{X}} \bar{p}_{\omega(i-1)z} v(z)}{v(\omega(i))}$$
$$= \prod_{i=1}^{n_\omega} \frac{\Theta(\epsilon^{\bar{d}(\omega(i-1),g)})}{\Theta(\epsilon^{\bar{d}(\omega(i),g)})} = \frac{\Theta(\epsilon^{\bar{d}(s,g)})}{\Theta(\epsilon^{\bar{d}(\omega(n_\omega),g)})} = \Theta(\epsilon^{d(s,g)}).$$

The second equality follows directly from Equation (9), the third equality from the lemma's assumption and Lemma 4.1, and the last equality from Lemma 4.2.

The second more specific result follows by observing that since the set Λ is finite and contains no high-probability cycles, there is an upper bound on how much likelihood ratio can be accumulated between two transitions of ϵ-order ≥ 1. □

Lemma 4.4. *If $v(x) = \Theta(\epsilon^{\bar{d}(x,g)})$ uniformly in x, then with $\mathbb{Q}(\omega)$ according to Equation (9), we have for any real-valued $k \geq 1$*

$$\mathbb{E}_{\mathbb{Q}}(L_{\mathbb{Q}}^k \cdot \mathbf{1}_\Phi) = \Theta(\epsilon^{kd(s,g)}).$$

Proof. Start by calculating an upper bound on the k'th moment (see below for explanation):

$$\begin{aligned}
\mathbb{E}_{\mathbb{Q}}(L_{\mathbb{Q}}^k \cdot \mathbf{1}_\Phi) &= \sum_{\omega\in\Phi(s)} \mathbb{Q}(\omega) \left(\frac{\mathbb{P}(\omega)}{\mathbb{Q}(\omega)}\right)^k = \sum_{r=d(s,g)}^{\infty} \sum_{\omega\in\Phi^r(s)} \mathbb{P}(\omega) \left(\frac{\mathbb{P}(\omega)}{\mathbb{Q}(\omega)}\right)^{k-1} \\
&\leq \sum_{r=d(s,g)}^{\infty} \sum_{\omega\in\bar{\Phi}^r(s)} \mathbb{P}(\omega) \left(\frac{\mathbb{P}(\omega)}{\mathbb{Q}(\omega)}\right)^{k-1} \\
&\leq \sum_{r=d(s,g)}^{\infty} \sum_{\omega\in\bar{\Phi}^r(s)} \mathbb{P}(\omega) \left(c_0^r \epsilon^{d(s,g)}\right)^{k-1} \\
&= \epsilon^{d(s,g)\cdot(k-1)} \sum_{r=d(s,g)}^{\infty} \mathbb{P}(\bar{\Phi}^r) \left(c_0^r\right)^{k-1} \leq c_1 \epsilon^{kd(s,g)},
\end{aligned} \tag{10}$$

where c_0 and c_1 are positive constants, and $\bar{\Phi}^r$ is like Φ^r, but with paths ending at their first visit to $\{g\} \cup \Gamma$ rather than at g. Since paths reaching or passing through Γ have at least ϵ-order $d(s,g)+1$ by definition of Γ, it follows that for any path $\omega \in \cup_{r\geq d(s,g)+1}\Phi^r(s)$, the path $\omega' = (\omega_0, \omega_1, \ldots, \omega_{m(\omega)})$ is in $\cup_{r\geq d(s,g)+1}\bar{\Phi}^r(s)$, with $m(\omega)$ as defined in Equation (8); and together with $\mathbb{P}/\mathbb{Q} = 1$ for steps on a path beyond Γ, this motivates the first inequality. The second inequality follows from Lemma 4.3. The third inequality is established by observing that $\mathbb{P}(\bar{\Phi}^r(s)) = \Theta(\epsilon^r)$, which is not trivial, since an infinite number of subdominant paths could conceivably contribute more than something that is $\Theta(\epsilon^r)$, but the bound follows from the finiteness of and the absence of HPCs in $\Lambda \cup \Gamma$, and a geometric series argument as used in the proof of Theorem 1 of L'Ecuyer and Tuffin (2011) (and in Lemma 5.6 of Reijsbergen (2013)).

A lower bound on the k'th moment is found by restricting the summation to only the dominant paths:

$$\mathbb{E}_{\mathbb{Q}}(L_{\mathbb{Q}}^k \cdot \mathbf{1}_\Phi) \geq \sum_{\omega\in\Delta(s)} \mathbb{Q}(\omega) \left(\frac{\mathbb{P}(\omega)}{\mathbb{Q}(\omega)}\right)^k \geq c_2 \epsilon^{kd(s,g)} \sum_{\omega\in\Delta(s)} \mathbb{Q}(\omega) \geq c_3 \epsilon^{kd(s,g)}, \tag{11}$$

where the last equality uses Lemma 4.3, in essence saying that under $\mathbb{Q}$, the dominant paths have total probability $\Theta(1)$. In Equation (11), c_2 and c_3 are positive constants. □

Theorem 4.5. *If $v(x) = \Theta(\epsilon^{\bar{d}(x,g)})$ uniformly in x, then the estimator based on v and $\mathbb{Q}$ according to Equation (9) has BRE:*

$$\frac{\mathrm{Var}_{\mathbb{Q}}(L_{\mathbb{Q}} \cdot \mathbf{1}_{\Phi})}{\mathbb{E}^2_{\mathbb{Q}}(L_{\mathbb{Q}} \cdot \mathbf{1}_{\Phi})} = O(1).$$

Proof. Immediate by using $\mathrm{Var}(X) = \mathbb{E}X^2 - \mathbb{E}^2X$ and applying Lemma 4.4. □

Theorem 4.6. *If $v(x) = \sum_{\omega \in \Delta(x)} \bar{\mathbb{P}}(\omega)$, then the estimator based on v and $\mathbb{Q}$ according to Equation (9) has VRE:*

$$\lim_{\epsilon \downarrow 0} \frac{\mathrm{Var}_{\mathbb{Q}}(L_{\mathbb{Q}} \cdot \mathbf{1}_{\Phi})}{\mathbb{E}^2_{\mathbb{Q}}(L_{\mathbb{Q}} \cdot \mathbf{1}_{\Phi})} = 0.$$

Proof. By the same argument as in Equation (10), we compute, for some positive c_4 and any real-valued $k \geq 1$,

$$\mathbb{E}_{\mathbb{Q}}(L^k_{\mathbb{Q}} \cdot \mathbf{1}_{\Phi \setminus \Delta}) = \sum_{r=1+d(s,g)}^{\infty} \sum_{\omega \in \Phi^r(s)} \mathbb{Q}(\omega) \left(\frac{\mathbb{P}(\omega)}{\mathbb{Q}(\omega)}\right)^k \leq c_4 \epsilon^{1+kd(s,g)} = O(\epsilon^{1+kd(s,g)}) = \epsilon \cdot O(v^k(s)).$$

Furthermore,

$$\begin{aligned}
\mathbb{E}_{\mathbb{Q}}(L^k_{\mathbb{Q}} \cdot \mathbf{1}_{\Delta}) &= \sum_{\omega \in \Delta(s)} \mathbb{P}(\omega) \left(\frac{\mathbb{P}(\omega)}{\mathbb{Q}(\omega)}\right)^{k-1} = \sum_{\omega \in \Delta(s)} \mathbb{P}(\omega) \left(\prod_{i=1}^{n_\omega} \frac{\sum_{z \in \mathcal{X}} \bar{p}_{\omega(i-1)z} v(z)}{v(\omega(i))}\right)^{k-1} \\
&= \sum_{\omega \in \Delta(s)} \mathbb{P}(\omega) \left(\prod_{i=1}^{n_\omega} \frac{v(\omega(i-1))}{v(\omega(i))} (1 + O(\epsilon))^{n_\omega}\right)^{k-1} \\
&= \sum_{\omega \in \Delta(s)} \mathbb{P}(\omega) \left(\frac{v(s)}{v(g)}\right)^{k-1} \cdot (1 + O(\epsilon)) = v^k(s) \cdot (1 + O(\epsilon)).
\end{aligned}$$

The second equality uses Equation (9), noting that for dominant paths $n_\omega = m(\omega)$; the third equality uses the fact that $v(z)$ is the sum of the dominant paths; and in the fourth equality $(1 + O(\epsilon))^{n_\omega} = 1 + O(\epsilon)$ is justified because n_ω is finite, as it is bounded from above by the maximum length of a dominant path through the finite set of states Λ. Comparing the above two results, we see that the contribution of the dominant paths dominates for all moments of the estimator. Hence,

$$\frac{\mathrm{Var}_{\mathbb{Q}}(L_{\mathbb{Q}} \cdot \mathbf{1}_{\Phi})}{\mathbb{E}^2_{\mathbb{Q}}(L_{\mathbb{Q}} \cdot \mathbf{1}_{\Phi})} = \frac{\mathbb{E}_{\mathbb{Q}}(L^2_{\mathbb{Q}} \cdot \mathbf{1}_{\Phi})}{\mathbb{E}^2_{\mathbb{Q}}(L_{\mathbb{Q}} \cdot \mathbf{1}_{\Phi})} - 1 = O(\epsilon).$$

Note that we cannot simply invoke Theorem 1 from L'Ecuyer and Tuffin (2011), because we have changed the model outside Λ. □

Theorem 4.7. *If $v(x) = \Theta(\epsilon^{\bar{d}(x,g)})$ uniformly in x and if the estimator based on v and $\mathbb{Q}$ according to Equation (9) does **not** have vanishing relative error, then it has the Bounded Normal Approximation (BNA) property:*

$$\frac{\mathbb{E}_{\mathbb{Q}}(|L_{\mathbb{Q}} \cdot \mathbf{1}_{\Phi} - \mathbb{E}_{\mathbb{Q}}(L_{\mathbb{Q}} \cdot \mathbf{1}_{\Phi})|^3)}{(\mathrm{Var}_{\mathbb{Q}}(L_{\mathbb{Q}} \cdot \mathbf{1}_{\Phi}))^{\frac{3}{2}}} = O(1).$$

Proof. Observe that, in general, for any positive a and b, it holds that $|a - b|^3 \leq (a + b)^3 = a^3 + 3a^2b + 3ab^2 + b^3$. Applying this to the numerator, we find it is upper-bounded by the sum of four expectation terms, each of which is of order $O(\epsilon^{3d(s,g)})$ by Lemma 4.4, so the same holds for the numerator as a whole.

For the denominator we find, again using Lemma 4.4,

$$\mathrm{Var}_{\mathbb{Q}}(L_{\mathbb{Q}} \cdot \mathbf{1}_{\Phi}) = \mathbb{E}_{\mathbb{Q}}(L_{\mathbb{Q}}^2 \cdot \mathbf{1}_{\Phi}) - \mathbb{E}_{\mathbb{Q}}^2(L_{\mathbb{Q}} \cdot \mathbf{1}_{\Phi}) = O(\epsilon^{2d(s,g)}).$$

If the variance does not vanish (condition of the theorem), the latter O can be replaced by Θ, completing the proof. □

Corollary 4.8. *ZVA-Δ has VRE and ZVA-$\bar{d}$ has BRE.*

Proof. ZVA-Δ uses $v(x) = v^{\Delta}(x)$ from Equation (7), which by definition (and by construction in the algorithms of Section 3.2) satisfies the requirement of Theorem 4.6. Similarly, ZVA-$\bar{d}$ uses $v(x) = \epsilon^{\bar{d}(x)}$, which clearly satisfies the requirement of Theorem 4.5. □

5 VARIANCE REDUCTION FOR FREE?

As part of the Path-ZVA algorithm we compute $v^{\Delta}(s)$, the probability of the dominant paths from s to g after HPC removal. When we run the simulation, we implicitly estimate this probability again through the sampling of dominant paths, which affects the estimator variance. Hence, we will explore the possibility of achieving further variance reduction for the estimator $\hat{\pi}$ by using this by-product of the numerical part of the algorithm. As before, let $\Phi = \Phi(s)$, and let $\Delta = \Delta(s)$, $\Psi = \Phi \setminus \Delta$, and $\mathbb{P}$ the probability measure after HPC removal. In words, Ψ is the set of paths that are not dominant but which still contribute to the probability of interest. We will discuss two variations: one in which $\mathbb{P}(\Delta)$ is used, and one in which we also compute $\mathbb{Q}(\Delta)$.

In the first variation, we use the fact that we already know $\mathbb{P}(\Delta)$ by ignoring all runs in which a dominant path is sampled. To see how this is done, note that

$$\mathbb{P}(\Phi) = \mathbb{P}(\Delta) + \mathbb{P}(\Psi) = \mathbb{P}(\Delta) + \mathbb{E}_{\mathbb{Q}}(L_{\mathbb{Q}} \cdot \mathbf{1}_{\Psi}). \tag{12}$$

If we only estimate the final expectation in Equation (12), we obtain the following estimator:

$$\hat{\pi}^{+} \triangleq \mathbb{P}(\Delta) + \frac{1}{N} \sum_{i=1}^{N} L_{\mathbb{Q}}(\omega_i) \cdot \mathbf{1}_{\Psi}(\omega_i). \tag{13}$$

This is equivalent to setting to zero all likelihood ratios obtained from the sampling of dominant paths, and adding $\mathbb{P}(\Delta)$ to the final result.

In the second variation, we also compute $\mathbb{Q}(\Delta)$ by running the same procedure that we used for $\mathbb{P}(\Delta)$, but under the new measure. We then use the fact that

$$\begin{aligned} \mathbb{P}(\Phi) &= \mathbb{P}(\Delta) + \mathbb{E}_{\mathbb{Q}}(L_{\mathbb{Q}} \cdot \mathbf{1}_{\Psi}) \\ &= \mathbb{P}(\Delta) + \mathbb{E}_{\mathbb{Q}}(L_{\mathbb{Q}} | \Psi) \cdot \mathbb{Q}(\Psi). \end{aligned} \tag{14}$$

Although we have not explicitly computed $\mathbb{Q}(\Psi)$, it holds under ZVA that $\mathbb{Q}(\Phi) = 1$ because transitions to t are given probability zero. Hence, $\mathbb{Q}(\Psi) = 1 - \mathbb{Q}(\Delta)$. In practice, we again generate samples $\omega_1, \ldots, \omega_N$, but if ω_i is not in Ψ we *discard* it, giving rise to the alternative sample $\omega'_1, \ldots, \omega'_M$ where M is the number of samples that are not in Δ. The resulting estimator is given by

$$\hat{\pi}^{++} \triangleq \mathbb{P}(\Delta) + \mathbb{Q}(\Psi)\, Y \text{ with } Y = \begin{cases} \frac{1}{M} \sum_{i=1}^{M} L_{\mathbb{Q}}(\omega'_i) & \text{if } M > 0 \\ 0 & \text{if } M = 0. \end{cases} \tag{15}$$

The separate treatment of $M = 0$ is needed to avoid division by zero, but does not affect the consistency of the estimator. Note that we do not need to multiply $L_{\mathbb{Q}}(\omega'_i)$ by $\mathbf{1}_{\Phi}(\omega'_i)$ because $\mathbb{Q}(\Phi) = 1$.

Next, let us calculate the variance of $\hat{\pi}^{++}$:

$$\begin{aligned}
\mathrm{Var}_{\mathbb{Q}}(\hat{\pi}^{++}) &= \mathrm{Var}_{\mathbb{Q}}\left(\mathbb{P}(\Delta) + \mathbb{Q}(\Psi)Y\right) = \mathbb{Q}(\Psi)^2\mathrm{Var}_{\mathbb{Q}}Y \\
&= \mathbb{Q}(\Psi)^2(\mathbb{E}_{\mathbb{Q}}\mathrm{Var}_{\mathbb{Q}}(Y|M) + \mathrm{Var}_{\mathbb{Q}}\mathbb{E}_{\mathbb{Q}}(Y|M)) \\
&= \mathbb{Q}(\Psi)^2\mathbb{E}_{\mathbb{Q}}\begin{Bmatrix} \frac{1}{M}\mathrm{Var}_{\mathbb{Q}}(L_{\mathbb{Q}}|\Psi) & \text{if } M>0 \\ 0 & \text{if } M=0 \end{Bmatrix} + \mathbb{Q}(\Psi)^2\mathrm{Var}_{\mathbb{Q}}\begin{Bmatrix} \mathbb{E}_{\mathbb{Q}}(L_{\mathbb{Q}}|\Psi) & \text{if } M>0 \\ 0 & \text{if } M=0 \end{Bmatrix} \\
&\approx \mathbb{Q}(\Psi)^2\frac{\mathrm{Var}_{\mathbb{Q}}(L_{\mathbb{Q}}|\Psi)}{N\mathbb{Q}(\Psi)} = \frac{\mathbb{Q}(\Psi)}{N}\mathrm{Var}_{\mathbb{Q}}(L_{\mathbb{Q}}|\Psi),
\end{aligned}$$

where the second line uses the law of total variance, and the approximation in the fourth line is the limit for $N \to \infty$. This limit is motivated by observing that M has a binomial distribution with parameters N and $\mathbb{Q}(\Psi)$, which becomes increasingly peaked around its mean $N\mathbb{Q}(\Psi)$ as $N \to \infty$.

Next, decompose the variance of the original importance sampling estimator $\hat{\pi}$:

$$\begin{aligned}
\mathrm{Var}_{\mathbb{Q}}(\hat{\pi}) &= \frac{1}{N}\mathrm{Var}_{\mathbb{Q}}(L_{\mathbb{Q}} \cdot \mathbf{1}_{\Phi}) \\
&= \frac{1}{N}\left(\mathbb{E}_{\mathbb{Q}}[\mathrm{Var}_{\mathbb{Q}}(L_{\mathbb{Q}} \cdot \mathbf{1}_{\Phi}|\mathbf{1}_{\Delta})] + \mathrm{Var}_{\mathbb{Q}}[\mathbb{E}_{\mathbb{Q}}(L_{\mathbb{Q}} \cdot \mathbf{1}_{\Phi}|\mathbf{1}_{\Delta})]\right) \\
&= \frac{1}{N}\mathbb{Q}(\Psi) \cdot \mathrm{Var}_{\mathbb{Q}}(L_{\mathbb{Q}} \cdot \mathbf{1}_{\Phi}|\Psi) + \frac{1}{N}\mathbb{Q}(\Delta) \cdot \mathrm{Var}_{\mathbb{Q}}(L_{\mathbb{Q}} \cdot \mathbf{1}_{\Phi}|\Delta) + \frac{1}{N}\mathrm{Var}_{\mathbb{Q}}[\mathbb{E}_{\mathbb{Q}}(L_{\mathbb{Q}} \cdot \mathbf{1}_{\Phi}|\mathbf{1}_{\Delta})], \\
&\approx \mathrm{Var}_{\mathbb{Q}}(\hat{\pi}^{++}) + \frac{1}{N}\mathbb{Q}(\Delta) \cdot \mathrm{Var}_{\mathbb{Q}}(L_{\mathbb{Q}} \cdot \mathbf{1}_{\Phi}|\Delta) + \frac{1}{N}\mathrm{Var}_{\mathbb{Q}}[\mathbb{E}_{\mathbb{Q}}(L_{\mathbb{Q}} \cdot \mathbf{1}_{\Phi}|\mathbf{1}_{\Delta})].
\end{aligned}$$

The latter two terms in this equation are variances and, hence, positive, meaning that $\hat{\pi}$ will (for large N) have larger variance than $\hat{\pi}^{++}$. This will be demonstrated using a case study in Section 6.1.3.

6 EXPERIMENTAL RESULTS

In this section, we present the results of simulation experiments with the Path-ZVA method. The aim of the experiments is twofold. In Sections 6.1, we focus on illustrative examples meant to demonstrate theoretical results and elucidate core concepts, namely, the BRE and VRE properties (Section 6.1.1), the nature of Λ and Γ in a practical example (Section 6.1.1), HPC removal (Section 6.1.2), and the performance of $\hat{\pi}^{+}$ and $\hat{\pi}^{++}$ (Section 6.1.3). In Section 6.2, we demonstrate the good performance of the new method using several realistic models from the literature. We compare it to the BFB and IGBS methods discussed in Section 2.3, and to the results for two case studies presented by Carrasco (2006). All of the experiments were conducted using a general framework written in Java, and the code needed to run the experiments is available on http://datashare.is.ed.ac.uk/handle/10283/2630. All experiments involve a particular class of models, namely, highly reliable *multicomponent systems*. Although this is already a very broad class of models, we emphasise that our procedure works for any HRMS (see the sample models included with the algorithm's code for several other applications, such as a 2-node tandem queue).

The simulation methods that we consider are standard Monte Carlo (MC), BFB and IGBS from the literature, and two variations of our ZVA method, namely, ZVA-$\bar{d}$ and ZVA-Δ as defined in Section 3.1.

6.1 Illustrative Examples

6.1.1 A Basic Example. Our first example is a multicomponent system with two component types, and k_1 and k_2 components of types 1 and 2, respectively. The system states are denoted by (x_1, x_2), in which $x_i, i \in \{1, 2\}$, is the number of components of type i that have failed. For each component type, one component is active at each time, with the other components acting

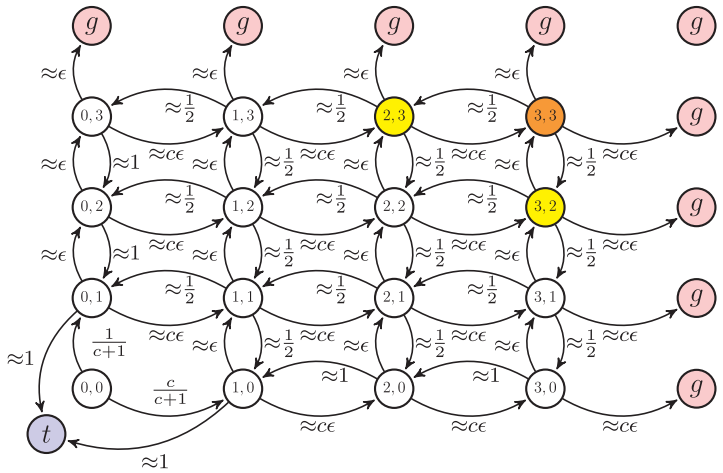

Fig. 3. Example of the case study of Section 6.1.1, with $k_1 = k_2 = 4$. The blue state is t, the pink states (marked g) are to be merged into a single state g, the yellow states (2,3) and (3,2) form Γ, the states in $\Lambda \setminus \{g, t\}$ are white, and $\mathcal{X} \setminus (\Lambda \cup \Gamma)$ in this case contains only state (3,3), coloured orange.

Table 3. Confidence Intervals (95%) for π as Functions of ϵ for the Different Simulation Methods, for the Model of Figure 3

ϵ	MC	BFB	ZVA-$\bar{d}$	ZVA-Δ
0.1	$1.038{\cdot}10^{-3}$ ± 6.08%	$9.955{\cdot}10^{-4}$ ± 1.00%	$9.994{\cdot}10^{-4}$ ± 0.12%	$9.999{\cdot}10^{-4}$ ± 0.10%
0.01	—	$1.010{\cdot}10^{-6}$ ± 1.46%	$1.000{\cdot}10^{-6}$ ± 0.04%	$1.000{\cdot}10^{-6}$ ± 0.03%
0.001	—	$9.940{\cdot}10^{-10}$ ± 1.55%	$1.000{\cdot}10^{-9}$ ± 0.01%	$1.000{\cdot}10^{-9}$ ± 0.01%
1.0E-4	—	$1.014{\cdot}10^{-12}$ ± 1.54%	$1.000{\cdot}10^{-12}$ ± 0.00%	$1.000{\cdot}10^{-12}$ ± 0.00%

Sample size: 10,000 runs.

as spares. The rate at which the active component of type 1 fails equals $c\epsilon$, $c \in (0, \infty)$, while the active component of type 2 fails with rate ϵ. Each component type has a dedicated repair unit which begins work immediately after the first component has failed, and which repairs a single component with a rate of 1. The system as a whole fails if all components of at least one of the two types have failed. Both the initial state s and regeneration state t are (0,0); as usual, we are interested in the probability of reaching a failure state g before returning to (0,0).

A DTMC is created for this model (and all other models in this section) by assigning to transitions from x to z, with $x, z \in \mathcal{X}$, a probability equal to the rate of transitions from x to z divided by the total exit rate of state x. A graphical representation of such a DTMC is given in Figure 3 for $k_1 = k_2 = 4$. The model has no HPCs, and, depending on k_1 and k_2, the dominant paths are given by the two straight paths from $(0, 0)$ to $(k_1, 0)$ and $(0, k_2)$. If $k_1 = k_2$, both paths are dominant, otherwise the shortest path is the unique dominant path. It holds that $d(s, g) = \min(k_1, k_2)$, and a state (x_1, x_2) is in Λ iff $x_1 + x_2 \leq \min(k_1, k_2)$.

In Table 3, we present a summary of a basic simulation experiment with different values of ϵ for each of the main simulation methods discussed in this article, performed on the model with $k_1 = k_2 = 4$. It can be seen that ZVA does much better than the other methods for sufficiently small values of ϵ. We expect VRE for ZVA-Δ and BRE for ZVA-$\bar{d}$ by Corollary 4.8, which is indeed confirmed by the table.

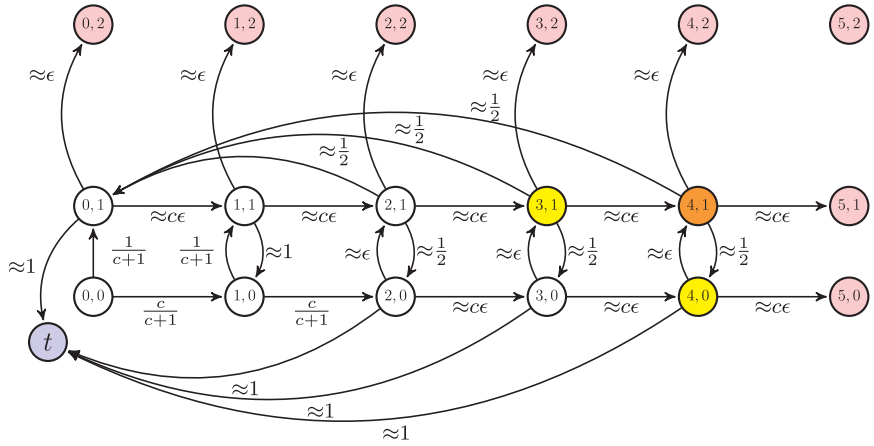

Fig. 4. Same system as in Figure 3, except with $k_1 = 5$ and $k_2 = 2$. Also, component type 1 is subject to deferred group repair, meaning that repair starts when two components have failed, and all components are repaired at the same time. The colouring is the same as in Figure 3.

6.1.2 Group/Deferred Repair. We now discuss the impact of HPCs on the performance of the various importance sampling methods. HPCs can emerge naturally in a multicomponent system if repair strategies are used that cause repairs to be slow or inactive in certain states of the system. It is known that BFB does not do well when HPCs are present; to remedy this, a more intricate version of BFB has been proposed, called IGBS (Juneja and Shahabuddin 2001); see Section 2.3 for more details. In this section, we will see that BFB will not do well in this setting, and IGBS only in some cases depending on the choice of parameters.

The setting that we consider first is depicted as a DTMC in Figure 4. Here, $k_1 = 5$ and $k_2 = 2$, and the repair strategy for component type 1 includes both deferred and group repair. Deferred repair means that the repair unit for component type 1 will not begin work until a minimum number of components have broken down—two in this case. Group repair means that when repair has begun, all components are repaired at the same time. The DTMC contains an HPC between states $(1, 0)$ and $(1, 1)$. This has a large impact on the dominant paths. Specifically, one dominant path is the path $((0, 0), (0, 1), (0, 2))$, which occurs with probability $\frac{1}{c+1}\epsilon$. The other dominant paths are those that jump from $(0, 0)$ to $(1, 0)$, then cycle between $(1, 0)$ and $(1, 1)$ k times, $k \in \{0, 1, \ldots\}$, and then jump to $(1, 2)$. These paths have a total probability contribution of

$$\frac{c}{(c+1)^2} \cdot \sum_{i=0}^{\infty} \left(\frac{1}{c+1} \cdot \frac{1}{1+(c+1)\epsilon} \right)^i \cdot \epsilon \;=\; \frac{1}{c+1}\epsilon + o(\epsilon),$$

so for small ϵ, roughly one half of the total probability mass is contributed by the path going to $(0, 2)$ and the other half by the ones going to $(1, 2)$.

During the pre-processing step, the HPC is detected by Algorithm 1 when the transition from state $(1, 1)$ to state $(1, 0)$ is considered. At that point, state $(1, 0)$ has already been determined to be in Λ, whilst the "cost" of reaching these states in terms of ϵ-orders is the same. Hence, the condition in line 8 is satisfied, which triggers the HPC removal procedure of Algorithm 2. Note that for the model of Figure 3, a HPC is (correctly) not detected because the "cost" to reach states $(1, 0)$ and $(1, 1)$ is different. The set of states in the HPC in Figure 4 (i.e., the set L of line 17) equals $\{(1, 0), (1, 1)\}$. This means that all transitions within L are removed and the probabilities of ending up in states $(2, 0)$, $(2, 1)$, and (1.2) from the two states in L are determined via line 19. For example,

Table 4. Confidence Intervals (95%) for π as a Function of ϵ for the Model of Figure 4, with $c = \frac{1}{50}$

ϵ	MC	BFB	IGBS	ZVA-$\bar{d}$	ZVA-Δ
0.1	$1.133{\cdot}10^{-1} \pm 5.48\%$	$9.865{\cdot}10^{-2} \pm 6.63\%$	$8.933{\cdot}10^{-2} \pm 26.5\%$	$1.065{\cdot}10^{-1} \pm 1.92\%$	$1.063{\cdot}10^{-1} \pm 0.01\%$
0.001	$1.300{\cdot}10^{-3} \pm 54.3\%$	$1.166{\cdot}10^{-3} \pm 19.9\%$	$2.241{\cdot}10^{-3} \pm 20.5\%$	$1.892{\cdot}10^{-3} \pm 6.61\%$	$1.912{\cdot}10^{-3} \pm 0.00\%$
1.0E-5	—	$1.180{\cdot}10^{-5} \pm 19.7\%$	$1.718{\cdot}10^{-5} \pm 23.9\%$	$2.038{\cdot}10^{-5} \pm 6.78\%$	$1.960{\cdot}10^{-5} \pm 0.00\%$
1.0E-7	—	$1.140{\cdot}10^{-7} \pm 6.96\%$	$1.921{\cdot}10^{-7} \pm 23.5\%$	$2.019{\cdot}10^{-7} \pm 6.79\%$	$1.961{\cdot}10^{-7} \pm 0.00\%$

Here, $\pi \approx 100/51 \cdot \epsilon \approx 1.96\epsilon$. Sample size: 10,000 runs. A "—" means that the rare event was not observed at all.

Table 5. Confidence Intervals (95%) for π as a Function of ϵ for the Model of Figure 4 with Two Changes: $(k_1, k_2) = (5, 3)$, and for Components of Type 1, the First Fails with Rate ϵ and the Others with Rate ϵ^2

ϵ	BFB	IGBS	ZVA-$\bar{d}$	ZVA-Δ
0.1	$2.692{\cdot}10^{-2} \pm 46.0\%$	$9.391{\cdot}10^{-2} \pm 128\%$	$4.651{\cdot}10^{-2} \pm 1.30\%$	$4.644{\cdot}10^{-2} \pm 1.02\%$
0.01	$3.440{\cdot}10^{-4} \pm 47.0\%$	$6.125{\cdot}10^{-3} \pm 62.4\%$	$4.939{\cdot}10^{-3} \pm 0.44\%$	$4.961{\cdot}10^{-3} \pm 0.33\%$
0.001	$2.933{\cdot}10^{-6} \pm 57.1\%$	$1.348{\cdot}10^{-4} \pm 29.2\%$	$4.995{\cdot}10^{-4} \pm 0.14\%$	$4.987{\cdot}10^{-4} \pm 0.13\%$
1.0E-4	$6.664{\cdot}10^{-8} \pm 99.9\%$	$1.797{\cdot}10^{-6} \pm 41.2\%$	$4.998{\cdot}10^{-5} \pm 0.05\%$	$5.000{\cdot}10^{-5} \pm 0.03\%$
1.0E-5	$4.093{\cdot}10^{-10} \pm 57.9\%$	$3.654{\cdot}10^{-8} \pm 79.9\%$	$4.999{\cdot}10^{-6} \pm 0.02\%$	$5.000{\cdot}10^{-6} \pm —$

Sample size: 10,000 runs. Note that ZVA-$\bar{d}$ has VRE because of the specific simple structure of v^{Δ} in this model. A confidence interval width of "—" means that no variance was observed.

for $\epsilon = \frac{1}{100}$, this leads to probabilities of roughly 66%, 0.6%, and 33% of reaching states $(2, 0)$, $(2, 1)$, and $(1, 2)$, respectively, from state $(1, 0)$.

BFB will not do well for small values of c; a cycle occurs with a probability of roughly $1/(c + 1)$ under $\mathbb{P}$, which is close to one if c is close to zero, but BFB will only assign probability $\frac{1}{4}$ to these cycles. This means that the dominant paths that contain many cycles will be sampled infrequently, resulting either in underestimation (see Devetsikiotis and Townsend (1993)) when these paths are not sampled in a simulation experiment, or high relative errors if they are sampled as each cycle blows up the likelihood ratio roughly by a factor $4/(c + 1) \approx 4$. IGBS mitigates the impact of this phenomenon by setting the probability of each HPC to δ^2 instead of $\frac{1}{4}$, with $\delta^2 < \frac{1}{4}$. Still, "good" choices of δ depend on c, so this requires a non-trivial knowledge of the system. This is illustrated in Table 4, in which BFB can be seen to suffer from underestimation (as witnessed by, e.g., its confidence interval not containing the true value of approximately $1.960 \cdot 10^{-5}$ for $\epsilon = 10^{-5}$). Note that the confidence interval bounds in the first columns do not seem trustworthy, probably because we have too few samples and/or very large fourth moments. By contrast IGBS (with $\delta = \frac{1}{100}$) is accurate in the sense that its confidence interval contains the true value, although it does not perform as well as ZVA.

As the value of c, and therefore the probability of leaving the HPC, is decreased, the performance of IGBS will worsen. In the extreme case where the probability of leaving the HPC decreases proportionally with ϵ, this is particularly visible. Consider the following modifications to the previous example: k_1 now equals 3, and the failure rate for components of type 1 is ϵ for the first component and ϵ^2 for the spare components. In Table 5, we have displayed the results for this setting. Here, IGBS does not contain the true value of approximately $\frac{1}{2}\epsilon$ for smaller values of ϵ. We have not included standard MC because of the very large amount of time it takes to sample runs.

6.1.3 Variance Reduction for Free. Table 6 shows a comparison of the different estimators discussed in Section 5, using the model of Figure 4. We see that $\hat{\pi}^{++}$ has notably better performance than the standard estimator, whereas $\hat{\pi}^{+}$ is worse. The difference between $\hat{\pi}^{++}$ and $\hat{\pi}$ varies between models—e.g., for the model of Figure 3 their performance is roughly the same, and in some

Table 6. Confidence Intervals (95%) for π as Functions of ϵ for the Different Simulation Methods Discussed in Section 5, for the Model of Figure 4

ϵ	$\hat{\pi}$	$\hat{\pi}^+$	$\hat{\pi}^{++}$	N	M	$N\mathbb{Q}(\Delta)$
0.1	$1.063{\cdot}10^{-1} \pm 0.0092\%$	$1.063{\cdot}10^{-1} \pm 0.2327\%$	$1.063{\cdot}10^{-1} \pm 0.0018\%$	10,000	141	143
0.01	$1.622{\cdot}10^{-2} \pm 0.0060\%$	$1.621{\cdot}10^{-2} \pm 0.1161\%$	$1.622{\cdot}10^{-2} \pm 0.0001\%$	10,000	35	42
0.001	$1.912{\cdot}10^{-3} \pm 0.0009\%$	$1.912{\cdot}10^{-3} \pm 0.0392\%$	$1.912{\cdot}10^{-3} \pm —$	10,000	4	5
1.0E-4	$1.956{\cdot}10^{-4} \pm 0.0001\%$	$1.956{\cdot}10^{-4} \pm —$	$1.9556{\cdot}10^{-4} \pm —$	10,000	0	1

models we have even observed $\hat{\pi}^{++}$ performing worse than $\hat{\pi}^+$. However, as is evident from Table 6, the potentially minor cost of performing the numerical pre-processing step a second time can lead to a reduction in confidence interval width of over 75% (e.g., see the row for $\epsilon = 0.01$). We will consider the pre-processing runtimes in more detail in the next section. Note that when no non-dominant paths are drawn (i.e., $M = 0$), $\hat{\pi}^{++}$ is no longer able to produce an estimate of the estimator variance and $\hat{\pi}$ is to be preferred.

6.2 Realistic Examples

In this section, we demonstrate the good performance of the Path-ZVA approach using two models from the literature. The first is the Distributed Database System, a classic literature benchmark that has been studied since the 1970s (Rosenkrantz et al. 1978), but which remains relevant today. In Section 6.2.1, we study the variation from Boudali et al. (2008) and Reijsbergen et al. (2010), and use Path-ZVA to compare the performance of different repair strategies. In Section 6.2.2, we study the variation from Carrasco (2006), and the fault-tolerant computing system of the same paper.

Instead of π, the probability of reaching the goal set during a regeneration cycle (i.e., before returning to the taboo state), the probability of interest in Carrasco (2006) is the system *unavailability*, denoted here by $\mathbf{v}$. It is defined as the steady-state probability of being in the goal set. We will also consider this measure in this section in order to compare results. We use $\mathbf{v} = \mathbb{E}(Z)/\mathbb{E}(D)$, where D is the total duration of a regeneration cycle (i.e., time between two visits to the taboo state) and Z is the amount of time spent in the goal set during a regeneration cycle. Typically, we estimate $\mathbb{E}(D)$ using standard MC, while $\mathbb{E}(Z)$ is estimated based on an estimate of π. For a more elaborate discussion, see, e.g., Reijsbergen et al. (2010). Note that HPC removal does have non-trivial consequences for state sojourn times and hence estimates for $\mathbf{v}$, although this does not affect the case studies because they do not have HPCs.

Additionally, each table now also displays the runtimes and Work-Normalized Variance Ratios (WNVRs) with respect to standard MC. We use the following WNVR definition: For a method m, let w_m be its confidence interval half-width and ρ_m the total time needed to produce the result. Then, the WNVR for this method is given by $(w_{\text{MC}}/w_m)^2 \cdot \rho_{\text{MC}}/\rho_m$. The WNVR represents the fact that to reduce the confidence interval width by a factor c, one would need to draw c^2 as many samples. It allows for easy comparison between methods with different runtimes; higher values of the WNVR indicate better performance.

6.2.1 The Distributed Database System. In this variant, the system consists of 9 component types: one set of 2 processors, two sets of 2 controllers each, and 6 disk clusters, with 6 disks each (see Figure 5). The failure rates for individual components are $\epsilon^2/2$ for processors and disk controllers, and $\epsilon^2/6$ for disks. The rates of component repairs are 1 for processors and disk controllers, and ϵ for disks. Note that the ϵ-orders are not part of the benchmark setting: the disk repairs being asymptotically slower than the other repairs is specific to this article. An interpretation would be that the data on the disks needs to be replicated, whereas the processors and the disk controllers

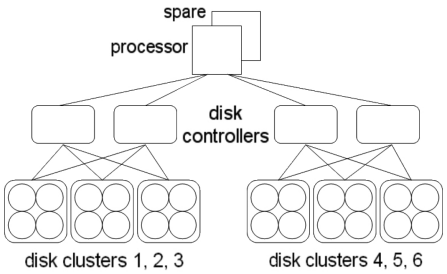

Fig. 5. Distributed database system.

only require hardware replacement. If we had assigned the same ϵ-order to the repairs of each of the types, then the four repair strategies would have the exact same asymptotic performance. The total failure rate for each component type depends linearly on the number of working components of that type; e.g., four working disks in disk set 1 means a total failure rate of $2\epsilon^2/3$ for disk set 1. The system as a whole is down if both processors are down, if both disk controllers in one of the controller sets are down, or if four disks are down in a single cluster. Both s and t are the state where all components are up. We consider four repair strategies:

(1) A dedicated repair unit for each of the nine component types.
(2) One repair unit, with priority given to high component type indices (i.e., disks first, then controllers, then processors).
(3) One repair unit, with priority given to low component type indices (i.e., processors first, then controllers, then disks).
(4) One repair unit, with a First Come First Served (FCFS) policy.

From a modelling point of view, Strategy 4 is the least tractable; to keep track of the order in which the components failed, a vector representing the number of failed components of each type is not sufficient. Specifically, if k components are down, then there are $k!$ ways in which this could have happened chronologically. This poses two problems. First, the size of the state space blows up dramatically, from 421,875 to 2,123,047,371 states. Second, if a modelling language is used that does not support lists (e.g., PRISM's reactive modules language), even a high-level description of the model can be hard to give. However, in the Java framework that we use for the experiments, states that contain lists are not conceptually harder to implement than vectors. The sizes of the sets Λ and Γ for the four strategies are as follows: 155 and 399 for dedicated repair, 561 and 448 for disk priority, 175 and 463 for processor priority, and 578 and 4,428 for FCFS. In all cases, $\Lambda \cup \Gamma$ is much smaller than the full state space.

In Table 7, we compare the four repair strategies in terms of their performance. Disk priority and FCFS are much more failure prone than the other strategies, because system failure due to two processors or disk controllers breaking becomes more likely if the repair unit is working on a disk. Apart from ZVA-Δ, we also present results obtained using the model checking tool PRISM, which approximates the probability of interest using numerical techniques (e.g., Gauss-Seidel) applied to the transition probability matrix. We see that our methods are accurate, albeit less efficient than PRISM, which was typically able to find the probability of interest within a second.

6.2.2 Fault-Tolerant Control/Database Systems. Two models are presented by Carrasco (2006, Section VI): the Fault-Tolerant Database System (FTD) and the Fault-Tolerant Control System (FTC). The FTD is a variation of the Distributed Database System discussed in the previous

Table 7. Confidence Intervals (95%) as a Function of ϵ, Generated Using Path-ZVA; Comparison of Repair Strategies for the DDS with Slow Disk Repairs

Model	ϵ	Estimate (π)	N	Runtime (ms)		PRISM
				Num.	Sim.	
DDS, ded. rep.	0.1	$2.997{\cdot}10^{-3} \pm 6.81\%$	69,272	1,663	10,000	$3.441{\cdot}10^{-3}$
	0.03	$1.802{\cdot}10^{-4} \pm 1.71\%$	815,322	1,501	10,000	$1.859{\cdot}10^{-4}$
	0.01	$1.786{\cdot}10^{-5} \pm 1.43\%$	1,522,273	1,420	10,000	$1.790{\cdot}10^{-5}$
	0.003	$1.543{\cdot}10^{-6} \pm 1.71\%$	1,495,413	1,440	10,000	$1.532{\cdot}10^{-6}$
DDS, disk prior.	0.1	$4.466{\cdot}10^{-2} \pm 16.6\%$	4,361	1,117	10,003	$4.925{\cdot}10^{-2}$
	0.03	$1.634{\cdot}10^{-3} \pm 6.92\%$	14,205	1,041	10,000	$1.704{\cdot}10^{-3}$
	0.01	$1.356{\cdot}10^{-4} \pm 2.54\%$	33,536	1,042	10,002	$1.367{\cdot}10^{-4}$
	0.003	$1.099{\cdot}10^{-5} \pm 0.66\%$	103,258	1,056	10,000	$1.101{\cdot}10^{-5}$
DDS, proc. prior.	0.1	$8.642{\cdot}10^{-3} \pm 13.4\%$	3,292	790	10,007	$8.419{\cdot}10^{-3}$
	0.03	$1.944{\cdot}10^{-4} \pm 6.18\%$	19,236	783	10,000	$1.954{\cdot}10^{-4}$
	0.01	$1.797{\cdot}10^{-5} \pm 2.51\%$	54,301	787	10,001	$1.798{\cdot}10^{-5}$
	0.003	$1.520{\cdot}10^{-6} \pm 0.87\%$	1,61,383	789	10,000	$1.533{\cdot}10^{-6}$
DDS, FCFS	0.1	$3.058{\cdot}10^{-2} \pm 22.1\%$	1,278	38,116	10,007	—
	0.03	$1.384{\cdot}10^{-3} \pm 16.9\%$	3,512	40,400	10,004	—
	0.01	$1.304{\cdot}10^{-4} \pm 6.14\%$	5,068	41,954	10,002	—
	0.003	$1.066{\cdot}10^{-5} \pm 2.50\%$	7,483	38,669	10,000	—

section—the goal and taboo sets are the same. The FTD has 10 component types; however, there are two types of failures so the state is represented using a 20-dimensional vector. Additionally, the model has failure propagation: a failure of a processor of the first type may trigger a failure of a processor of the second type. There are two parameter settings (I and II). In setting I the system is "balanced" in the sense that the ϵ-orders of all failure transitions equals 1, whereas in setting II some failures have ϵ-order 1 and others ϵ-order 2. The second model, the FTC, consists of 39 component types, and system failure is a non-trivial function of the state. Because of space constraints, we refer the reader to Carrasco (1992) or our programming code for a full description of the model. We only consider the first out of four possible parameter settings for the FTC. The technique proposed in the paper, called Balanced Failure Transition Distance Biasing (BFTDB), is a refinement of the method proposed by Carrasco (1992) to ensure good performance for unbalanced systems.

As we can see from Table 8, Path-ZVA has a roughly similar performance to BFTDB, which is to be expected since they are based on the same principles. BFTDB does slightly better than Path-ZVA for the FTC because of the relatively large probability contribution of paths that leave Λ—the numerical procedure behind BFTDB determines $\bar{d}$ for all states, which means that it is able to perform better in this specific setting. (Note that their numerical approach cannot be applied to general HRMSs, e.g., those that include HPCs). BFB does not perform well in our implementation because it draws much fewer samples per second than the other schemes. This is because we use the default biasing probability of 0.5 for failures, which means that a typical sample path will be considerably longer than under the other schemes. For example, under MC the sample path will typically reach the taboo state very quickly, whereas under Path-ZVA the system quickly reaches the goal state or a state outside Λ after which IS is turned off.

Note that in all models the transition rates are fixed, so the choice of ϵ is arbitrary. As we discussed in Section 3.3, our approach is to fix a value ϵ and choose the ϵ-orders of the transitions as the smallest order such that the pre-factor $p_{xz}/\epsilon^{r_{xz}}$ is still greater than ϵ. The ϵ-values chosen by Carrasco (2006) were 0.00072 for both settings of the FTD and 0.00028 for setting A of the FTC.

Table 8. Comparison with Three Reliability Models from the Literature; BFTDB is the Simulation Method Proposed in Carrasco (2006)

Model	Method	Estimate ($\mathbf{v}$)	N	Runtime (ms)		WNVR
				Num.	Sim.	
FTD (I)	MC	$1.827{\cdot}10^{-8} \pm 8.10\%$	32,184,109	0	120,294	1.00
	BFB	$1.931{\cdot}10^{-8} \pm 14.9\%$	27,396	0	120,026	0.30
	ZVA-$\bar{d}$	$1.827{\cdot}10^{-8} \pm 0.23\%$	2,554,961	5,994	120,005	1,178.93
	ZVA-Δ	$1.829{\cdot}10^{-8} \pm 0.19\%$	2,726,637	4,755	120,001	1,847.67
	BFTDB	$1.814{\cdot}10^{-8} \pm 0.23\%$	2,999,000	0	112,000	268.22
FTD (II)	MC	$1.728{\cdot}10^{-8} \pm 6.51\%$	42,956,445	0	120,001	1.00
	BFB	$1.636{\cdot}10^{-8} \pm 12.5\%$	28,372	0	120,005	0.27
	ZVA-$\bar{d}$	$1.622{\cdot}10^{-8} \pm 0.20\%$	3,110,348	522	120,001	1,011.29
	ZVA-Δ	$1.621{\cdot}10^{-8} \pm 0.22\%$	2,756,305	527	120,001	888.40
	BFTDB	$1.621{\cdot}10^{-8} \pm 0.15\%$	3,999,000	0	150,000	284.54
FTC (A)	MC	$3.232{\cdot}10^{-10} \pm 50.7\%$	20,827,971	0	120,001	1.00
	BFB	$1.908{\cdot}10^{-10} \pm 77.8\%$	3,008	0	120,052	0.43
	ZVA-$\bar{d}$	$2.682{\cdot}10^{-10} \pm 0.94\%$	827,848	448,936	120,048	609.53
	ZVA-Δ	$2.696{\cdot}10^{-10} \pm 0.38\%$	937,337	478,349	120,054	3,627.10
	BFTDB	$2.694{\cdot}10^{-10} \pm 0.15\%$	1,204,000	0	319,000	8,112.75

For the FTD we used $\epsilon = 0.001$, and for the FTC we used $\epsilon = 0.00072$. We use a confidence level of 95%, whereas Carrasco (2006) uses 99%; the results have been rescaled accordingly. Runtimes for BFTDB are from Carrasco (2006), while the other runtimes are from our tool; because of software implementation and hardware differences, the comparison is at best indicative. For our experiments, we chose the simulation runtime in each case to be around 2 minutes, leading to different numbers of runs for the different methods.

We have observed that using an ϵ-value of 0.001 for the FTD led to a large reduction in terms of the size of $\Lambda \cup \Gamma$ and hence the duration of the pre-processing step, without adversely affecting the performance of Path-ZVA to a notable extent. This is what we have used for Table 8.

7 CONCLUSIONS

We have introduced a rare event simulation method that is generally applicable to HRMSs, provided that the relevant subset Λ is numerically tractable. This is often the case, but not always, e.g., when the reliability of the system is due to high component redundancy. We have mathematically proved its efficiency and discussed an automated implementation. We have demonstrated its good performance across a range of case studies, including a realistic benchmark model. For one repair strategy (FCFS), the new method was able to compute probabilities that cannot be obtained using either standard Monte Carlo or the numerical approximation techniques used in, e.g., PRISM. We also discussed a further variance reduction technique and demonstrated its good performance. The code for the experiments is available on http://datashare.is.ed.ac.uk/handle/10283/2630 for download.

There are several directions for future work. The simulation code has not been optimised for performance, so improving it is future work. The variance reduction technique of Section 5 could be studied in more detail, and across a wider range of models. Finally, we could compare the performance of our method to a wider range of other IS techniques, e.g., the cross-entropy method.

ACKNOWLEDGMENTS

The authors would like to thank Jane Hillston for her helpful comments on a draft version of this article.

REFERENCES

E. Ábrahám, N. Jansen, R. Wimmer, J. P. Katoen, and B. Becker. 2010. DTMC model checking by SCC reduction. In *Proceedings of the 7th International Conference on the Quantitative Evaluation of Systems (QEST)*. IEEE, 37–46.

C. Alexopoulos and B. C. Shultes. 2001. Estimating reliability measures for highly-dependable Markov systems, using balanced likelihood ratios. *IEEE Transactions on Reliability* 50, 3 (2001).

J. Barnat, J. Chaloupka, and J. van de Pol. 2011. Distributed algorithms for SCC decomposition. *Journal of Logic and Computation* 21, 1 (2011), 23–44.

H. Boudali, P. Crouzen, B. R. Haverkort, M. Kuntz, and M. Stoelinga. 2008. Arcade—A formal, extensible, model-based dependability evaluation framework. In *Proceedings of the 13th IEEE International Conference on Engineering of Complex Computer Systems*, Vol. 3. IEEE, 243–248.

C. E. Budde, P. R. D'Argenio, and H. Hermanns. 2015. Rare event simulation with fully automated importance splitting. In *European Workshop on Performance Engineering*. Springer, 275–290.

M. Capiński and P. E. Kopp. 2004. *Measure, Integral and Probability*. Springer.

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

J. A. Carrasco. 2006. Failure transition distance-based importance sampling schemes for the simulation of repairable fault-tolerant computer systems. *IEEE Transactions on Reliability* 55, 2 (2006), 207–236.

D. R. Cox. 1962. *Renewal Theory*. Methuen & Co., London.

P. T. de Boer, P. L'Ecuyer, G. Rubino, and B. Tuffin. 2007. Estimating the probability of a rare event over a finite time horizon. In *Proceedings of the 2007 Winter Simulation Conference*. IEEE, 403–411.

M. Devetsikiotis and J. K. Townsend. 1993. An algorithmic approach to the optimization of importance sampling parameters in digital communication system simulation. *IEEE Transactions on Communications* 41, 10 (1993), 1464–1473.

E. W. Dijkstra. 1959. A note on two problems in connexion with graphs. *Numerische Mathematik* 1, 1 (1959), 269–271.

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

S. Juneja. 2007. Estimating tail probabilities of heavy tailed distributions with asymptotically zero relative error. *Queueing Systems* 57, 2 (2007), 115–127.

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

M. Kwiatkowska, G. Norman, and D. Parker. 2011. PRISM 4.0: Verification of probabilistic real-time systems. In *Computer Aided Verification*. Springer, 585–591.

A. M. Law and W. D. Kelton. 1991. *Simulation Modeling and Analysis*. McGraw-Hill, New York.

P. L'Ecuyer, J. Blanchet, B. Tuffin, and P. Glynn. 2010. Asymptotic robustness of estimators in rare-event simulation. *ACM Transactions on Modeling and Computer Simulation (TOMACS)* 20, 1 (2010), 6.

P. L'Ecuyer and B. Tuffin. 2008. Approximate zero-variance simulation. In *Proceedings of the 2008 Winter Simulation Conference*. IEEE, 170–181.

P. L'Ecuyer and B. Tuffin. 2011. Approximating zero-variance importance sampling in a reliability setting. *Annals of Operations Research* 189, 1 (2011), 277–297.

E. E. Lewis and F. Böhm. 1984. Monte Carlo simulation of Markov unreliability models. *Nuclear Engineering and Design* 77, 1 (1984), 49–62.

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

D. Reijsbergen. 2013. *Efficient Simulation Techniques for Stochastic Model Checking*. Ph.D. dissertation. University of Twente, Enschede.

D. Reijsbergen, P. T. de Boer, W. Scheinhardt, and B. R. Haverkort. 2010. Rare event simulation for highly dependable systems with fast repairs. In *Proceedings of the 7th International Conference on the Quantitative Evaluation of Systems (QEST'10)*. IEEE, 251–260.

D. Reijsbergen, P. T. de Boer, W. Scheinhardt, and B. R. Haverkort. 2013. Automated rare event simulation for stochastic Petri nets. In *Proceedings of the 10th International Conference on the Quantitative Evaluation of Systems (QEST'13)*. Springer, 372–388.

A. Ridder. 2010. Asymptotic optimality of the cross-entropy method for Markov chain problems. *Procedia Computer Science* 1, 1 (2010), 1571–1578.

D. J. Rosenkrantz, R. E. Stearns, and P. M. Lewis II. 1978. System level concurrency control for distributed database systems. *ACM Transactions on Database Systems (TODS)* 3, 2 (1978), 178–198.

P. Shahabuddin. 1994. Importance sampling for the simulation of highly reliable Markovian systems. *Management Science* 40 (1994), 333–352.

Received November 2015; revised November 2017; accepted November 2017