INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

A Simulation Method for Network Performability Estimation using Heuristically-computed Pathsets and Cutsets

Franco Robledo — Pablo Sartor

N° 8267

Mars 2013

Domaine 3

Rapport technique

ISSN 0249-0803 ISRN INRIA/RT--8267--FR+ENG

A Simulation Method for Network Performability Estimation using Heuristically-computed Pathsets and Cutsets

Franco Robledo, Pablo Sartor

Domaine : Réseaux, systèmes et services, calcul distribué
Équipes-Projets Dionysos

Rapport technique n° 8267 — Mars 2013 — 21 pages

Abstract: Consider a set of terminal nodes K that belong to a network whose nodes are connected by links that fail independently with known probabilities. We introduce a method for estimating any performability measure that depends on the hop distance between terminal nodes. It generalises previously introduced Monte Carlo methods for estimation of the K-reliability of networks with variance reduction compared to crude Monte Carlo. They are based on using sets of edges named d-pathsets and d-cutsets for reducing the variance of the estimator. These sets of edges, considered as a priori known in previous literature, heaviliy affect the attained performance; we hereby introduce and compare a family of heuristics for their selection. Numerical examples are presented, showing the significant efficiency improvements that can be obtained by chaining the edge set selection heuristics to the proposed Monte Carlo sampling plan.

Key-words: heuristics, pathset, cutset, Monte Carlo, rare events, reliability, performability, bounded length, diameter constraints

Une méthode de simulation pour l'estimation de la performabilité des réseaux utilisant pathsets et cutsets calculés heuristiquement

Résumé : Considérez un ensemble de nœuds terminaux K appartenant à un réseau dont les nœuds sont reliés par des liaisons qui échouent indépendamment avec des probabilités connues. Nous présentons une méthode pour estimer n'importe quelle mesure de performabilité qui dépend de la distance en sauts entre les nœuds terminaux. Elle généralise des méthodes de Monte Carlo précédemment introduites pour l'estimation de la K-fiabilité des réseaux avec réduction de la variance par rapport à Monte Carlo standard. Ces méthodes sont basées sur l'utilisation d'ensembles d'arêtes designés d-pathsets et d-cutsets pour réduire la variance de l'estimateur. Ces ensembles d'arêtes, considérés comme connus a priori dans la littérature précédente, affectent fortement les performances atteintes ; nous introduisons et comparons une famille d'heuristiques pour leur sélection. Des exemples numériques sont présentés, montrant les importantes améliorations dans l'efficacité qui peuvent être obtenues par le chaînage de ces heuristiques avec le plan d'échantillonnage de Monte Carlo proposé.

Mots-clés : heuristiques, pathset, cutset, Monte Carlo, événements rares, fiabilité, performabilité, longueur bornée, restrictions de diamètre

1 Introduction

Consider a communication network whose components randomly fail, modelled as an undirected graph. The most classical reliability analysis model assigns to the network two possible states determined by the link states; it is operational if and only if a certain set of distinguished sites, known as terminals, are connected, otherwise it is failed. A generalization introduced in [12] imposes also an upper limit in the allowed distance between terminals for the network to be considered as operational. This generalization, conceived to model situations where limits exist in the acceptable delay times or the number of hops undergone by data packets, keeps the binary nature of the network state. But when in comes to performability, in several contexts there is need to employ metrics defined over a larger number of network states, characterised by the hop distance between terminals. For example, in voice-over-IP applications, the perceived quality is affected by latency, which is in turn determined by the number of links traversed by packets. Quality deteriorates as high hop-distance states occur more frequently in the network. In web applications with rich interfaces, the quality perceived by the end user is related to responsiveness, where latency determines the delay between the user actions and their effect on the output interface. The same consideration applies to other contexts where costs are born each time a link is traversed e.g. vehicle or packet routing, where costs can relate to time spent, tolls, fuel, etc.

Classical reliability analysis consists of computing, estimating or bounding the probability that the network is operational, that is, the expected value of the binary variable associated with the network state. Surveys can be found in [7], [15] and [13]. In particular, many Monte Carlo methods have been proposed for efficient estimation of this expected value. This article introduces a Monte Carlo method to estimate any performability metric that is defined as the expected value of a distance-dependent network metric. In real communication networks, link reliabilities are normally very high. Then, when applying simulation methods, sampling a 'high distance' network state is a rare event. In the context of Monte Carlo simulations for network reliability analysis, once fixed a certain confidence interval goal, the needed sample size unboundedly grows as link reliabilities become higher. Several variance-reduction techniques can be used to reduce the sample size; surveys can be found in [14], [3] and [9]; more recent works include [4], [5], [11], [16] and [2]. [10] and [8] introduced a family of methods for the classical reliability, based on sampling strategies conditioned by paths and cuts. In [6] it is shown how to extend them to include diameter constraints, employing sets of edges named d-pathsets and d-cutsets. The Monte Carlo method hereby proposed generalises these methods in order to estimate the expected value of a random variable determined by the maximal distance between pairs of terminals.

The d-pathsets and d-cutsets, that were considered as *a priori* known in the previously mentioned literature, heaviliy affect the performance attained by the simulation methods. We hereby introduce and compare a family of heuristics for their selection. We present numerical evidence of the significant efficiency gains attained by chaining these heuristics to the proposed simulation method when compared to a crude Monte Carlo simulation.

The remainder of the article is organised as follows. Section 2 includes definitions, notation and model formalisation. Section 3 describes the crude

Monte Carlo method and the estimators it gives for the metric under study as well as for its variance. Section 4 describes the suggested Monte Carlo method and shows the variance reductions achieved relative to the crude one. Section 5 describes the heuristics for selecting the (d-)pathsets and (d-)cutsets that will be applied in the simulation. Section 6 presents three numerical examples, based on mesh-like networks. It compares several variations of the heuristic, the relative efficiency of the proposed Monte Carlo method vs. the crude one and how it is influenced by link reliabilities. Finally, conclusions and further work are summarised in Section 7.

2 Definitions and Notation

The network is modelled by an undirected graph $G = (V, E)$ with $n = |V|$, $m = |E|$ and $E = \{e_1 ... e_m\}$, whose nodes and edges correspond to the sites and links of the network respectively. The following definitions and notation are also employed:

- $K \subseteq V$: a subset of nodes that corresponds to the distinguished sites (called *terminal nodes* or simply *terminals*);
- X_e: for every $e \in E$, a random binary variable whose value is 1 if e operates and 0 otherwise;
- r_e: reliability of e (the probability that it is operational at any given instant). The edges are assumed to fail independently of one another;
- $X = (X_1 ... X_m) \in \{0, 1\}^m$: a *network configuration* (an m-tuple encoding the states of all edges);
- $\mathcal{X}$: set of the 2^m possible network configurations;
- $\pi(x) = \Pr(X = x)$ (probability that the random network configuration is x);
- $\Delta : \mathcal{X} \to \{0, ..., n, \infty\}$: the function that gives the maximum distance $\Delta(x)$ between two terminals in the partial graph of G encoded by a network configuration x;
- K is *d-connected* in G if and only if, for each pair of nodes of K, there is a path between them, whose length is not above the integer d;
- Φ: network parameter to estimate (random variable determined by the network configurations).

Our goal is to estimate the expected value of the random variable Φ. The value of this random variable is determined by the network configurations as follows. The set $\{0, ..., n, \infty\}$ (the codomain of Δ) is partitioned into several intervals (think of them as 'quality levels'). Then each interval is mapped to one value from a set $\mathcal{Q} \subset \mathbb{R}$ with $|\mathcal{Q}| \leq r{+}1$. Therefore every network configuration x belongs to an unique 'quality level' that corresponds to a certain Φ value. Then, our final aim is to estimate the expected value of the function $\Phi : \mathcal{X} \to \mathcal{Q}$.

Figure 1 illustrates the model and notations used. Given r integers $0 < d_0 < \cdots < d_{r-1}$, let $\Delta = \{\Delta_0 \cup \Delta_1 \cup \cdots \cup \Delta_r\}$ be a set of intervals where

$\Delta_0 = (0, d_0]$, $\Delta_i = (d_{i-1}, d_i]$ for $i \in [1, r-1]$ and $\Delta_r = (d_{r-1}, \infty]$. Let $\mathcal{X}$ be partitioned (and its components called *regions*) as $\mathcal{X} = \mathcal{X}_0 \cup \cdots \cup \mathcal{X}_r$ where $\mathcal{X}_i = \{x \in \mathcal{X} \mid \Delta(x) \in \Delta_i\}$ for $i \in [0, r]$. Let p_i denote the probability that x is one of $\mathcal{X}_i$. Let also z_i denote the probability that x is one of $\mathcal{Z}_i$, being $\mathcal{Z}_i \subseteq \mathcal{X}_i$ defined in Section 4. Similarly, $\mathcal{Q} = \{\Phi_0, \Phi_1, \ldots, \Phi_r\}$ and $\Phi : \mathcal{X} \to \mathcal{Q}$ gets defined by $\Phi(x) = \Phi_i \iff \Delta(x) \in \Delta_i(x)$.

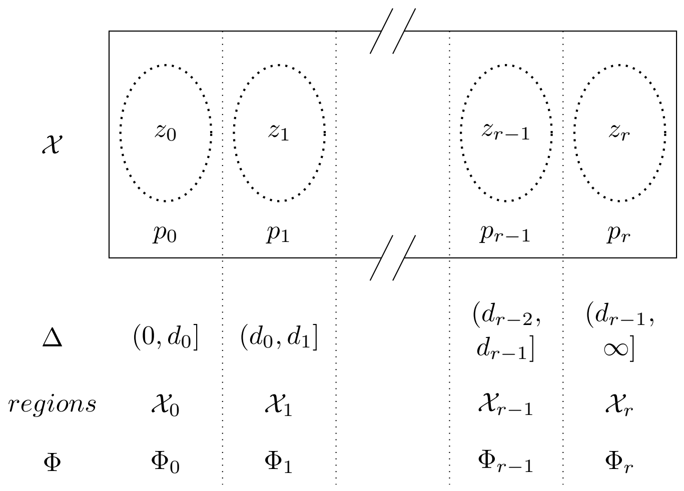

Figure 1: Partitions of the network configuration space

3 Crude Monte Carlo Method

A crude Monte Carlo simulation estimates the expected value $\bar{\Phi}$ of Φ by independently sampling N network configurations $x^{(1)}, \ldots, x^{(N)}$, computing Φ for each one, and building an estimator

$$\widehat{\Phi_N} = \frac{1}{N} \sum_{i=1..N} \Phi(x^{(i)})$$

whose variance is ($EV(\cdot)$ denotes the expected value)

$$\widehat{\sigma_N^2} = \frac{N}{N^2} var(\Phi) = \frac{1}{N}(EV(\Phi^2) - EV^2(\Phi)) = \frac{1}{N}(\sum_{i=1..N} \Phi_i^2 p_i - \bar{\Phi}^2).$$

Sampling each $x^{(i)}$ involves m Bernoulli trials for determining the state of each edge. Computing Φ for $x^{(i)}$ is done by applying a breadth-first-search (BFS) algorithm starting at every node of K. So each iteration takes a time that is $O(\max\{m, |K|^2\})$.

4 Proposed Monte Carlo Method

Suppose that certain topological knowledge about $G = (V, E)$ is available in the form of certain edge sets, called *pathsets*, *cutsets*, d_i*-pathsets* and d_i*-cutsets* for a given $i \in [0..r-1]$, that we define next. Let P be any subset of E and $G' = (V, P)$ the partial graph of G yielded by P. Then,

- P is a *pathset* if and only if K is connected in G';
- P is a *d-pathset* if and only if K is d-connected in G';
- a (d-)pathset is said to *operate* when all of its edges operate.

Similarly let C be any subset of E and $G' = (V, E \setminus C)$ the partial graph of G yielded by $E \setminus C$. Then,

- C is a *cutset* if and only if K is not connected in G';
- C is a *d-cutset* if and only if K is not d-connected in G';
- a (d-)cutset is said to *fail* when all of its edges are failed.

Hereafter, unless otherwise specified or clear by the context, the terms pathset and cutset refer to any of the above defined sets, regardless of the presence or absence of a length constraint. In our context, an elementary event is a network configuration X, with operational edges $\{e \in E/X_e = 1\}$ and failed edges $\{e \in E/X_e = 0\}$. The sets of operating/failed edges define whether a given pathset/cutset operates/fails.

Under certain circumstances, the simultaneous occurrence of operating/failed sets allows to know the value of Φ for a given network configuration. For example, suppose that a certain network configuration X is such that a given 5-pathset operates while a given 2-cutset fails. It follows that the maximal distance between the nodes of K must be any of $\{3, 4, 5\}$ in the partial graph encoded by X. If the interval $(2, 5]$ belongs to Δ, then the region to which X belongs is known, and so is its Φ value. The proposed method takes advantage of this property as we see next. Assume that the following sets of edges are known:

- $\mathcal{P}_0$: set of some pathsets;
- $\mathcal{P}_1, \dots, \mathcal{P}_{r-1}$: $r-1$ sets such that each $\mathcal{P}_i$ is a set of some d_i-pathsets;
- $\mathcal{P}_r = \emptyset$ (for convenience of notation);
- $\mathcal{C}_0 = \emptyset$ (for convenience of notation);
- $\mathcal{C}_1, \dots, \mathcal{C}_{r-1}$: $r-1$ sets such that each $\mathcal{C}_i$ is a set of some d_{i-1}-cutsets;
- $\mathcal{C}_r$: set of some cutsets.

In the previous definitions, the word 'some' means that every set can contain any number of elements ranging from zero to the maximum existing number of (d_i-)pathsets or (d_i)-cutsets. At least one of the sets must be non-empty for the method to be useful; if all sets are empty then the method coincides with crude Monte Carlo. Now, the following events can be defined over $\mathcal{X}$:

$$\begin{aligned}
\mathcal{T}_i &= (\text{some element of } \mathcal{P}_i \text{ operates}) && (i = 0, \cdots, r-1)\\
\mathcal{K}_j &= (\text{some element of } \mathcal{C}_j \text{ fails}) && (j = 1, \cdots, r)\\
\mathcal{Z}_0 &= \mathcal{T}_0 &&\\
\mathcal{Z}_h &= \mathcal{T}_h \wedge \mathcal{K}_h && (h = 1, \cdots, r-1)\\
\mathcal{Z}_r &= \mathcal{K}_r &&
\end{aligned}$$

With these definitions, given a network configuration x, it holds that $(\mathcal{Z}_0 \to (\Delta(x) \in (0, d_0]))$ and that $(\mathcal{Z}_i \to (\Delta(x) \in (d_{i-1}, d_i]))$ for $i \in [1 \ldots r]$. In other words, each event $\mathcal{Z}_{h=0\cdots r}$ determines a precise Φ value. The events $\mathcal{Z}_0, \ldots, \mathcal{Z}_r$ are subsets of $\mathcal{X}_0, \ldots, \mathcal{X}_r$ respectively and thus pairwise disjoint too. This is also shown in Figure 1, where $p_i = \Pr(x \in \mathcal{X}_i)$ and $z_i = \Pr(x \in \mathcal{Z}_i)$ for $i \in [0 \ldots r]$.

In what follows all summations have an implicit subscript $i = 0, \ldots, r$. Suppose that the probabilities $z_0, \ldots, z_r$ are easy to compute. Then it is easy to compute $\phi = \sum \Phi_i z_i$, the part of $\bar{\Phi}$ for which $\mathcal{Z}$ accounts. The method described below is based on computing the remaining part of $\bar{\Phi}$ (the one given by the events out of $\mathcal{Z}$) by restricting the sampling space to $\mathcal{X} \backslash \mathcal{Z}$. Let $z = \Pr(x \in \mathcal{Z}) = \sum z_i$. Our sampling plan estimates $\bar{\Phi}$ by sampling N network configurations $x^{(1)}, \ldots, x^{(N)}$ within $\mathcal{X} \backslash \mathcal{Z}$ (with a probability distribution that respects the relative probabilities among the network configurations in $\mathcal{X} \backslash \mathcal{Z}$), computing Φ for each one, and building an estimator $\widetilde{\Phi_N}$ with variance $\widetilde{\sigma_N^2}$ as follows:

$$\widetilde{\Phi_N} = \frac{1}{N} \left(\sum_i \Phi(x^{(i)})(1-z) \right) + \phi$$

whose variance is

$$\begin{aligned}
&\widetilde{\sigma_N^2} = \\
&\quad \frac{N}{N^2} var\left((1-z)\Phi\right) = \frac{(1-z)^2}{N}\left(EV(\Phi^2) - EV^2(\Phi)\right) = \\
&\quad \frac{(1-z)^2}{N}\left(\sum_i \frac{\Phi_i^2(p_i - z_i)}{1-z} - \left(\sum_i \frac{\Phi_i(p_i - z_i)}{1-z}\right)^2\right) = \\
&\quad \frac{1}{N}\left((1-z)\sum_i \Phi_i^2(p_i - z_i) - (\bar{\Phi} - \phi)^2\right).
\end{aligned}$$

Here the expected values involve probabilities conditioned to $\mathcal{X} \backslash \mathcal{Z}$, hence the application of the correction factor $1/(1-z)$ to $p_i - z_i$.

4.1 Variance Reduction

The variances obtained through the crude and the proposed Monte Carlo methods are next compared; for simplicity it is done for one single iteration (i.e. the simulation sample size is one). Single-index summations are over $i \in \{0, \ldots, r\}$ and double-index summations over all pairs $(i, j) \in \{0, \ldots, r\}^2$.

$$
\begin{aligned}
&\widehat{\sigma_1^2} - \widetilde{\sigma_1^2} = \\
&\quad \sum_i \Phi_i^2 p_i + (1-z)\sum_i \Phi_i^2 (p_i - z_i) - 2\bar{\Phi}\phi + \phi^2 = \\
&\quad z\sum_i \Phi_i^2 p_i - z\sum_i \Phi_i^2 z_i + \sum_i \Phi_i^2 z_i - \\
&\qquad 2(\sum_i \Phi_i p_i)(\sum_i \Phi_i z_i) + (\sum_i \Phi_i z_i)^2 = \\
&\quad \sum_{ij} \Phi_i^2 p_i z_j - \sum_{ij} \Phi_i^2 z_i z_j + \sum_{ij} \Phi_j^2 p_i z_j - \\
&\qquad 2\sum ij \Phi_i \Phi_j p_i z_j + \sum ij \Phi_i \Phi_j z_i z_j = \\
&\quad \sum_{ij} (\Phi_i - \Phi_j)^2 p_i z_j + \sum_{ij} (\Phi_i \Phi_j - \Phi_i^2) z_i z_j = \\
&\quad \sum_{ij} (\Phi_i - \Phi_j)^2 (p_i - z_i) z_j + \\
&\qquad \sum_{ij} \left((\Phi_i - \Phi_j)^2 + \Phi_i \Phi_j - \Phi_i^2 \right) z_i z_j = \\
&\quad \sum_{ij} (\Phi_i - \Phi_j)^2 (p_i - z_i) z_j + \sum_{ij} (\Phi_j(\Phi_j - \Phi_i)) z_i z_j = \\
&\quad \sum_{ij} (\Phi_i - \Phi_j)^2 (p_i - z_i) z_j + \sum_{i<j} (\Phi_j - \Phi_i)^2 z_i z_j .
\end{aligned}
$$

The following Lemma characterises the variance reduction in terms of the regions and their respective subsets determined by the set of pathsets and cutsets.

Lemma 1 *The difference of the variances $\widehat{\sigma_1^2} - \widetilde{\sigma_1^2}$ is always non-negative. Regarding strict positivity, it is a necessary and sufficient condition that two non-empty regions $\mathcal{X}_i$ and $\mathcal{X}_j$ exist such that their Φ values are different and $\mathcal{Z}_j$ is non-empty.*

Proof. Observe that both summations in the final form of expression of $\widehat{\sigma_1^2} - \widetilde{\sigma_1^2}$ only involve non-negative terms, hence the difference of variances is always non-negative. Regarding strict positivity, assume that two non-empty regions $\mathcal{X}_i$ and $\mathcal{X}_j$ exist such that $\Phi_i \neq \Phi_j$, $p_i > 0$ and $z_j > 0$ (therefore $p_j > 0$). Then, if $p_i > z_i$, the first summation will have a strictly positive term given by the subindices i, j. If $p_i = z_i$, then $z_i > 0$ and therefore the second summation will have a strictly positive term given by the subindices i, j. This proves the sufficiency of the statement about $\mathcal{X}_i$ and $\mathcal{X}_j$. Conversely, assume that $\widehat{\sigma_1^2} - \widetilde{\sigma_1^2} > 0$. Then there must exist i, j such that at least one of the corresponding terms in the first and second summation is strictly positive. If the term for the first summation is strictly positive, then $\Phi_i > \Phi_j$, $p_i > z_i$ and $z_j > 0$. Since $(p_i > z_i \rightarrow p_i > 0)$ and $(z_j > 0 \rightarrow p_j > 0)$ the statement about $\mathcal{X}_i$ and $\mathcal{X}_j$ holds true. If the term for the second summation is strictly positive, then $\Phi_i > \Phi_j$, $z_i > 0$ and $z_j > 0$. Again, $p_i > 0$ and $p_j > 0$ and the statement holds true. ♠

4.2 Sampling within $\mathcal{X} \setminus \mathcal{Z}$

Let $\Omega = \bigcup_{i=0 \dots r}(\mathcal{P}_i \cup \mathcal{C}_i)$ be the set of all edges occurring in at least one pathset or one cutset under consideration. Note that the event $\mathcal{Z}$ is independent from the state of all edges that do not belong to Ω. Then the state of the edges of $E \backslash \Omega$ can be easily sampled with $m - |\Omega|$ Bernoulli trials. A general sequential procedure to sample the state of the edges of Ω is the following. Let $(o_1 \dots o_{|\Omega|}) \in \{0,1\}^{|\Omega|}$ an $|\Omega|$-tuple encoding the sampled states. Assume that $o_1 \dots o_r$ have already been sampled (being $r < |\Omega|$) and let E_r be the event in which the first r edges of Ω have these sampled states respectively. Then the probability that the $r+1$-th edge of Ω is operational is (knowing that the edges of Ω must have states such that the network configuration belongs to $\mathcal{X} \setminus \mathcal{Z}$):

$$\Pr(o_{r+1} = 1 \mid \mathcal{X} \setminus \mathcal{Z} \wedge E_r) = \Pr(o_{r+1} = 1) \frac{\Pr(\mathcal{X} \setminus \mathcal{Z} \mid E_r \wedge o_{r+1} = 1)}{\Pr(\mathcal{X} \setminus \mathcal{Z} \mid E_r)}$$

where $\Pr(o_{r+1} = 1)$ is the reliability of the $r+1$-th edge of Ω. The function that given the reliabilities $\rho_1, \dots, \rho_{|\Omega|}$ of the edges of Ω returns the probability of the event $\mathcal{X} \setminus \mathcal{Z}$, is a polynomial $\mathbb{P}(\rho_1, \dots, \rho_{|\Omega|})$. So, computing $\Pr(\mathcal{X} \setminus \mathcal{Z} \mid E_r)$ involves replacing $\rho_1, \dots, \rho_r$ by 0 or 1 according to the states sampled for the first r edges and then evaluating $\mathbb{P}$. Finding (and evaluating) $\mathbb{P}$ can be very complex when pathsets and cutsets of the same region highly overlap. Observe that $\Pr(\mathcal{X} \setminus \mathcal{Z}) = 1 - \Pr(\mathcal{Z}) = 1 - \Pr(\bigwedge_i \mathcal{Z}_i) = 1 - \sum_i \Pr(\mathcal{T}_i \wedge \mathcal{K}_i)$ (recall the pairwise independence of all $\mathcal{Z}_i$). To compute each $\Pr(\mathcal{T}_i \wedge \mathcal{K}_i)$, if every edge involved occurs only in one of $\mathcal{P}_i$ and $\mathcal{C}_i$ then it is possible to get factorised expressions (see the Appendix in [6]), and then building and evaluating the polynomial can be done in time $O(|\Omega|)$. The previous considerations about the overlapping of edge sets within the same region also apply to computing $z_0, \cdots, z_r$.

For limited cardinalities of Ω an alternative approach can be used, consisting in precomputing the probability of the occurrence of each of the $2^{|\Omega|}$ possible sub-configurations that exist when only considering the edges of Ω, in $O(2^{|\Omega|})$ time. Then, sampling their states just involves choosing a sub-configuration at random through a cut-point access on a table accumulating the precomputed probabilities (thus in $O(|\Omega|)$ time). This is the fastest way to sample the states for the edges of Ω, but at the expense of the exponential-in-$|\Omega|$ effort for pre-computing the table, that can limit its applicability on large networks.

5 A heuristic for pathset and cutset generation

In this section we introduce a heuristic that generates pathsets and cutsets for every region $\mathcal{X}_i$. We develop an algorithm for the two-terminal problem that can be easily generalised to the problem for general sets K. For every i, we build a set of d_i-pathsets and d_{i-1}-cutsets, such that no two elements share edges. This set should ideally be the one that maximises the probability z_i that one d-pathset operates and one d-cutset fails at the same time, to attain the largest variance reduction that is possible. The basic idea consists of a greedy randomised generation of paths with lengths in $(d_{i-1}, d_i]$, followed by a greedy randomised generation of d_{i-1}-cutsets, without using any edge included in the former paths. The cycle is repeated several times and the combination of sets that yield the higher probability is finally chosen.

5.1 Paths generation

The algorithm shown in Fig. 2 receives the graph G, source and destination nodes (s, t) and the minimum and maximum distances ℓ_1, ℓ_2 that define the region. It returns a random path whose length is in the interval $[\ell_1, \ell_2]$ or the null $(\perp)$ element if no such path could be found. The algorithm proceeds with a greedy selection of nodes (`currNode`) that are added to the path under construction `newPath`. It starts by selecting s and ends after reaching t or achieving a point where it is impossible to complete a path with the required constraints. In each iteration, the shortest path (`shP`) between the current node and t is computed. If there is no such path, or its length exceeds the difference between ℓ_2 and the length of the already built part, the algorithm returns the null element $\perp$. Otherwise, the next edge can be either the first one of `shP`, or another one different from it. When there are more than one feasible edges, the first one of `shP` is chosen with a probability proportional to the ratio between the length of `shP` and the maximum number of edges that can be added to the path under construction without violating the $[\ell_1, \ell_2]$ constraint. The function `rand()` in the algorithm returns a random uniform real number in [0,1). Therefore the algorithm tends to stick to a shortest path when `newPath` has reached a length that leaves few chances to divert. On the contrary, when there is still room for many more edges than the length of a shortest path, the algorithm will tend to choose, at random, other directions to extend `newPath`. In line 11, care is taken to not repeat a node of `shP`, which would result in having cycles within the `newPath`. Lines 13-16 remove the chosen edge from G for future iterations, append it to `newPath` and update its length ℓ and `currNode`. Random is therefore introduced in the decision of wheter to make a step in the direction of a shortest path or not, as well as in the selection of the next node, in case it was decided not to follow the shortest path.

5.2 Cutsets generation

The algorithm for creating an ℓ-cutset given a certain integer ℓ is shown in Fig. 3. It receives the graph G, the source and terminal nodes (s, t), the integer ℓ and a set of edges H. It starts by building a first-in-first-out queue with all edges of $G \setminus H$ inserted in increasing order of distance to s. Random is introduced by shaking these distances prior to insertion. For example, in our tests we swaped each pair of values in $\vec{d}$ with a probability inversely proportional to their difference. The queue is later used to add each edge to the cutset under construction `newCut` in this shaken order; the idea behind this is that dropping edges in the vecinity of s is a good strategy to find low cardinality ℓ-cutsets for s, t. The set H will be used when invoking this algorithm to avoid using edges already used for other ℓ-pathsets or ℓ-cutsets found prior to the one under construction. The `while` loop proceeds adding edges to `newCut` and dropping them from G until one of the following occur: i) the distance between s and t is greater than ℓ (so we have an ℓ-cutset); or ii) the queue is empty (so no ℓ-cutset exists, which happens if and only if there was a path whose length is not above ℓ built exclusively with edges of $G \setminus H$). After the `while` loop, `newCut` is an

Procedure generatePath$(G, s, t, \ell_1, \ell_2)$

```
1: $\ell \leftarrow 0$; currNode $\leftarrow s$; newPath $\leftarrow \emptyset$
2: while currNode $\neq t$ do
3:   shP $\leftarrow$ shortestPath$(G,$ currNode$, t,$ newPath$)$
4:   dist $\leftarrow$ length(shP)
5:   if (shP $= \perp) \vee (\ell +$ dist $> \ell_2)$ then
6:     return $\perp$
7:   else
8:     if (currNode has only one neighbour in $G$) $\vee$ (rand() $<$ $(\ell +$ dist $- \ell_1)/(\ell_2 - \ell_1))$ then
9:       next $\leftarrow$ neighbour of currNode in shP
10:    else
11:      next $\leftarrow$ any neighbour of currNode in $G \setminus$ shp
12:    end if
13:    remove the edge (currNode,next) from $G$
14:    append the edge (currNode,next) to newPath
15:    currNode $\leftarrow$ next
16:    $\ell \leftarrow \ell + 1$
17:  end if
18: end while
19: return newPath
```

Figure 2: Heuristic algorithm for generating paths with lengths in $[\ell_1, \ell_2]$

ℓ-cutset not necessarily minimal (i.e. some edges can be dropped from it and still be an ℓ-cutset). The `for` loop builds an ℓ-cutset `minCut` that is minimal in this sense (although, in general, it will not be a minimum-cardinality ℓ-cutset).

5.3 The main heuristic

Our main algorithm iterates until a given amount of time is spent. In each iteration a disjoint set of edges for a certain region $\mathcal{X}_i$ is generated. Each iteration will begin with the generation of one d_i-pathset and one d_{i-1}-cutset. It then will continue adding more sets, following a certain sequence that defines a "version" of the algorithm. Each iteration may not exceed a certain parameter time MAX_TIME; if it does then it is discarded and a new iteration is run. For each generated sets P and C of d_i-pathsets and d_{i-1}-cutsets, the probability of the event z_i that they define is computed; the P,C pair with the highest $\Pr(z_i)$ is recorded as the algorithm proceeds and returned after timing out. The algorithm shown in Fig. 4 corresponds to the version where the sequence pathset-cutset-pathset-cutset is followed in each iteration; we will denote it as PCPC. It illustrates how to invoke the procedures `generatePath` and `generateCutset`. In Section 6.2 we compare the results obtained with seven different versions. The algorithm receives the graph G, source and destination nodes (s, t) and the range of distances allowed for the zone $[\ell_1, \ell_2]$. It returns the pair (P, C) whose probability was the highest among all the pairs generated. The pair is not necessarily built by two pathsets and two cutsets. Note the way that `generatePath` and `generateCutset` are invoked in lines 13 and 19. Passing

Procedure generateCutset(G, s, t, ℓ, H)

```
$\vec{d} \leftarrow$ vector of distances between $s$ and every node in $G \setminus H$
randomAlter($\vec{d}$)
queue $\leftarrow$ all edges of $G \setminus H$ inserted in increasing order of their values in $\bar{d}$
newCut $\leftarrow \emptyset$; flag $\leftarrow$ true
**while** flag **do**
  shP $\leftarrow$ shortestPath($G, s, t, \emptyset$)
  **if** (shP $= \perp$) $\vee$ (length(shP) $> \ell$) **then**
    flag $\leftarrow$ false
  **else if** isEmpty(queue) **then**
    **return** $\perp$
  **else**
    newEdge $\leftarrow$ pop(queue)
    remove newEdge from $G$
    add newEdge to newCut
  **end if**
**end while**
minCut $\leftarrow \emptyset$
**for all** $e \in$ newCut **do**
  add $e$ to $G$
  shP $\leftarrow$ shortestPath($G, s, t, \emptyset$)
  **if** (shP $\neq \perp$) $\wedge$ (length(shP) $\leq \ell$) **then**
    remove $e$ from $G$
    add $e$ to minCut
  **end if**
**end for**
**return** minCut
```

Figure 3: Heuristic algorithm for generating an ℓ-cutset

Procedure PCPC$(G, s, t, \ell_1, \ell_2)$

```
bestP ← $\perp$; bestC ← $\perp$; highestProb ← 0
while ellapsed_time < MAX_TIME do
  P ← $\emptyset$; C ← $\emptyset$
  repeat
    p ← generatePath($G, s, t, l_1, l_2$)
  until (p $\neq \perp$) or MAX_TRIES attempts were done
  if (p $= \perp$) then continue //aborts current "while" iteration
  P ← P $\cup$ {p}
  repeat
    c ← generateCutset($G, s, t, l_1 - 1$, P)
  until (c $\neq \perp$) or MAX_TRIES attempts were done
  if (c $= \perp$) then continue //aborts current "while" iteration
  C ← C $\cup$ {c}
  if Pr(P,C) > highestProb then
    highestProb ← Pr(P,C); bestP ← P; bestC ← C
  end if
  repeat
    p ← generatePath($G \setminus$ P, $s, t, l_1, l_2$)
  until (p $\neq \perp$) or MAX_TRIES attempts were done
  if (p $= \perp$) then continue //aborts current "while" iteration
  P ← P $\cup$ {p}
  if Pr(P,C) > highestProb then
    highestProb ← Pr(P,C); bestP ← P; bestC ← C
  end if
  repeat
    c ← generateCutset($G, s, t, l_1 - 1$, P $\cup$ C)
  until (c $\neq \perp$) or MAX_TRIES attempts were done
  if (c $= \perp$) then continue //aborts current "while" iteration
  C ← C $\cup$ {c}
  if Pr(P,C) > highestProb then
    highestProb ← Pr(P,C); bestP ← P; bestC ← C
  end if
end while
return bestP, bestC, highestProb
```

Figure 4: Pseudo-code for the main heuristic; version PCPC

$G \setminus$ P and P $\cup$ C allows respectively to obtain disjoint pathsets and cutsets respect to those so far generated in the current iteration. If a subprocedure is invoked MAX_TIMES times without succeeding to return a pathset or cutset, a new iteration is started. Similar algorithms PP, PPP, ... and CC, CCC, ... are used for the border regions $\mathcal{X}_0$ and $\mathcal{X}_r$, generating only pathsets or only cutsets (with no length constraint).

Δ	region	$r_e = 0.9$	$r_e = 0.95$	$r_e = 0.99$
up to 5	$\mathcal{X}_0$	0	0	0
6 to 7	$\mathcal{X}_1$	5	30	1,000
above 7	$\mathcal{X}_2$	10	60	2,000
disconnected	$\mathcal{X}_3$	20	120	4,000

Table 1: Test 1: fines per region.

6 Numerical examples

This section provides numerical examples based on mesh-like topologies. The simulations are inspired on the following situation. There is a contract between a communication network provider and a customer who needs to periodically exchange data between two sites s and t. They agree on a scale of fines, to be paid by the provider, according to the number of hops that each packet undergoes. The aim is to estimate the expected value of the fines that will be paid during the contract lifetime. To do so, simulations based on crude Monte Carlo and the proposed method were run and their results compared. The methods were implemented in `C++` and the tests were run on an Intel Core2 Duo T5450 machine with 2 GB of RAM, executing 10^7 iterations (sample size).

6.1 Test case 1 - ANTEL's transport network

Test case 1 is based on the countrywide transport network topology of ANTEL, the largest telecommunications provider in Uruguay, shown in Figure 5. Nodes represent the sites whose interfaces perform routing activity, thus adding significant latency. Links represent the existing paths between these sites. Three scenarios are considered, corresponding to interface failure probability values of 0.10, 0.05 and 0.01 respectively. The test illustrates the effect that the rarity of edge failures has on the attained efficiency relative to crude Monte Carlo. Assume that nodes 4 and 14 (shown as squares) are to be connected in a context where low latencies are desirable. Table 1 lists the scale of fines payed according to the hop distance between both nodes. Three scales are used, each one corresponding to a certain value of link reliability in the network model (0.90, 0.95 and 0.99). The simulations will estimate the expected value of the average fine as well as its variance. The fine scales were proportionaly adjusted so that the expected values of the fines to pay were rather similar, by running short simulations. Note how the fines per region must quickly increase to yield the same expected value of fines when link reliabilities become higher, due to the fact that network configurations with high distances, or disconnected, become rarer events. In other words, this means that the provider can agree on paying higher fines per region still facing the same fine expected value, because of improved link reliabilities. Table 2 shows the pathsets and cutsets employed, using the edge labels of Figure 5.

Table 3 shows $\hat{\Phi}$ and $\tilde{\Phi}$ (the expected values of Φ estimated by the crude and proposed methods respectively); the estimations obtained for $\widehat{\sigma^2}$ and $\widetilde{\sigma^2}$; and the total times (in seconds). As above mentioned the scale of fines was set up so that the expected values of the fines to pay were approximately the

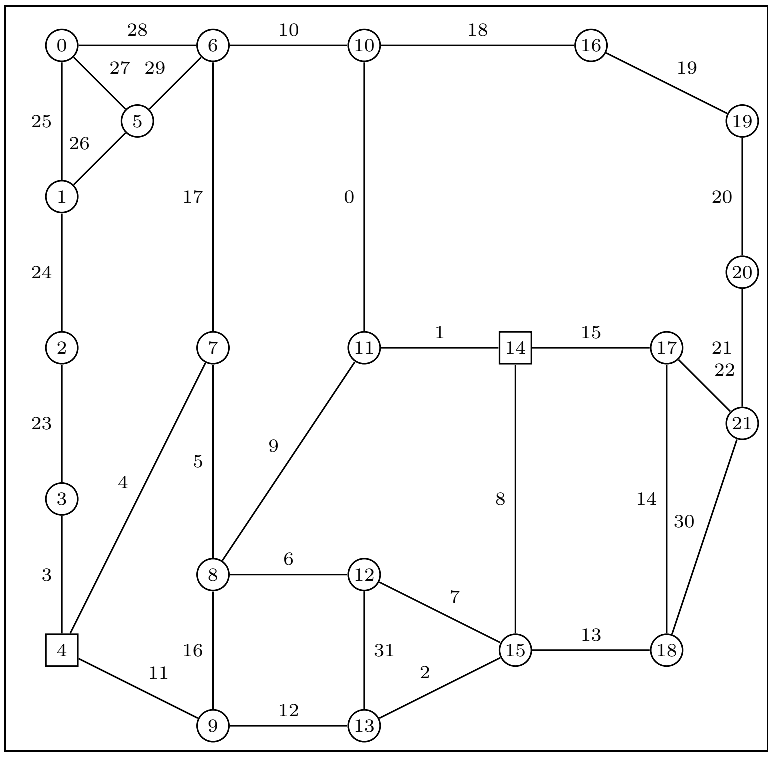

Figure 5: Test 2: reduced transport network topology of ANTEL, Uruguay's national telecommunications provider

reg.	$\mathcal{P}_i$	$\mathcal{C}_i$
$\mathcal{X}_0$	$\mathcal{P}_0 = \{\{4,5,9,1\},\{11,12,2,8\}\}$	$\mathcal{C}_0 = \emptyset$
$\mathcal{X}_1$	$\mathcal{P}_1 = \{\{11,12,2,13,14,15\}\}$	$\mathcal{C}_1 = \{\{1,8\}\}$
$\mathcal{X}_2$	$\mathcal{P}_2 = \{\{4,17,10,18,19,$ $20,21,22,15\}\}$	$\mathcal{C}_2 = \{\{1,8,13\}\}$
$\mathcal{X}_3$	$\mathcal{P}_3 = \emptyset$	$\mathcal{C}_3 = \{\{3,4,11\},\{1,8,15\}\}$

Table 2: Test 1: pathsets and cutsets.

same (they ranged from 0.341855 to 0.361926). Times spent are essentially the same across the three reliability scenarios, with the crude method taking a time approximately 11% lower than the proposed method. Observe the significant reductions achieved in the variance by the proposed method (13.63, 45.27 and 940.2 for r_e equal to 0.90, 0.95 and 0.99 respectively). An efficiency comparison can be reported via the *relative efficiency RE*, which is a standard ratio employed in simulation literature (see e.g. [8], [14]), defined as follows:

$$RE = \frac{var_{crude}}{var_{proposed}} \times \frac{time_{crude}}{time_{proposed}} \approx \frac{\widehat{\sigma^2}}{\widetilde{\sigma^2}} \times \frac{time_{crude}}{time_{proposed}}$$

.	Crude	Proposed	ratio
$r_e = 0.9$			
Φ	0.341855	0.341856	-
σ^2	4.538235×10^{-7}	3.328820×10^{-8}	13.63
t(s)	290.9	323.7	0.8987
$r_e = 0.95$			
Φ	0.361926	0.361632	-
σ^2	2.467931×10^{-6}	5.451282×10^{-8}	45.27
t(s)	285.8	321.3	0.8896
$r_e = 0.99$			
Φ	0.338000	0.334667	-
σ^2	6.018858×10^{-5}	6.401462×10^{-8}	940.2
t(s)	280.4	321.7	0.8715

Table 3: Test 1: numerical results.

Instance	Size	s	t	ℓ_1	ℓ_2	$\Pr(z_i)$
Grid1	8×8	(2,2)	(4,4)	6	10	1.895E-4
Grid2	8×8	(2,2)	(5,5)	8	12	1.974E-4
Grid3	15×15	(4,7)	(11,7)	10	20	2.529E-6
Grid4	15×15	(4,4)	(11,11)	18	24	9.575E-9

Table 4: Test 2: grid-based instances for heuristic tests.

The *RE* expresses the attained variance reduction adjusted by the spent-time ratio; in this case, its values are 12.25, 40.27 and 819.4 respectively for $r_e = 0.90$, $r_e = 0.95$ and $r_e = 0.99$.

6.2 Test 2 - Square grids

This test illustrates the behaviour of seven versions of the heuristic algorithm. One version (that we call PC) always returns one pathset and one cutset. Two versions (PCP, PCC) can return one extra pathset or cutset respectively. Finally, four versions (PCPP, PCPC, PCCP, PCCC) return two, three or four components whose nature corresponds to each letter. Two network topologies were employed: square grids with 8×8 and 15×15 nodes respectively. Table 4 shows the characteristics of the four instances of the problem that were run for each version of the algorithm. Nodes s and t are specified by their "x, y coordinates" in the grid (numbered from zero). The reliability of each edge was randomically set, according to a triangular distribution (0.985, 0.99, 0.995). The parameters MAX_TIME and MAX_TRIES were set to 40 seconds and 5 tries. Last column of the table shows the highest probability found among those returned by each version of the algorithm.

Table 5 shows the results of the four tests; within each one, the algorithms are sorted by descending order of the probability that each one returned. Column labelled %best reports the ratio between the returned probability and the highest one for that particular test. Column #edges reports the total number of edges involved in the pathsets and cutsets of the returned solution. Columns

	%best	#edges	2 sets	3 sets	4 sets	best (P,C)
Grid1						
PCCP	1.000	22	1,055	726	687	2,2
PCPC	0.998	24	671	637	255	2,2
PCCC	0.911	18	497	336	71	1,3
PCC	0.911	14	1,086	729	0	1,2
PCP	0.509	18	3,186	3,052	0	2,1
PCPP	0.501	18	2,635	2,518	943	2,1
PC	0.465	10	4,135	0	0	1,1
Grid2						
PCPC	1.000	28	417	401	152	2,2
PCCP	1.000	28	604	392	377	2,2
PCCC	0.898	20	463	299	266	1,3
PCC	0.898	16	605	387	0	1,2
PCP	0.502	22	2,012	1,907	0	2,1
PCPP	0.501	24	1,709	1,623	621	2,1
PC	0.458	12	2,439	0	0	1,1
Grid3						
PCCC	1.000	26	82	55	43	1,3
PCPC	0.772	40	79	61	30	2,2
PCCP	0.769	42	105	73	54	2,2
PCC	0.667	23	110	74	0	1,2
PCPP	0.394	56	286	196	59	3,1
PCP	0.386	37	310	204	0	2,1
PC	0.340	18	380	0	0	1,1
Grid4						
PCP	1.000	50	169	105	0	2,1
PCPP	0.996	52	180	98	64	2,1
PCPC	0.996	52	56	27	13	2,1
PCCP	0.988	62	31	21	12	2,2
PC	0.821	28	202	0	0	1,1
PCC	0.821	28	37	17	0	1,1
PCCC	0.821	28	28	17	12	1,1

Table 5: Test 2: ranking of results for the grid-based instances.

2s, 3s and 4s report the number of solutions generated that had two, three and four components (pathsets plus cutsets). Finally column best(P,C) reports the number of pathsets and cutsets in the solution returned by each algorithm. First, note that in all cases there is a significant difference between the best and worst returned solutions (their ratio ranging from 1,22 to 2,94). Second, three of the best solutions had four components and the remaining one had three. These results suggest that it might be worth to spend time looking for solutions with many components, rather than striving to get the best “one pathset - one cutset” possible solution, in topologies alike. Third, there is no clear winner, although PCCC, PCPC and PCCP seem to have the best overall results.

Δ	region	$r_e = 0.90$	$r_e = 0.95$	$r_e = 0.99$
up to 5	$\mathcal{X}_0$	0	0	0
6 to 12	$\mathcal{X}_1$	10	150	85,000
above 12	$\mathcal{X}_2$	1,000	15,000	8,500,000

Table 6: Test 3: fines per region.

.	Crude	Proposed	ratio
$r_e = 0.90$			
Φ	0.928540	0.928373	-
σ^2	9.135802×10^{-5}	2.591886×10^{-6}	35.25
t(s)	606.4	975.2	0.6218
$r_e = 0.95$			
Φ	0.933135	0.915608	-
σ^2	1.366002×10^{-3}	6.188060×10^{-6}	220.75
t(s)	586.2	950.4	0.6168
$r_e = 0.99$			
Φ	0.935000	0.928744	-
σ^2	7.232224×10^{-1}	3.217148×10^{-5}	22,480.23
t(s)	513.1	914.4	0.5612

Table 7: Test 3: numerical results.

6.3 Test 3 - Randomised extension of Arpanet

This test illustrates the combined effect of the heuristic algorithm for pathset and cutset generation and the variance-reduction technique. The network, shown in Fig 6, has 60 nodes and 110 edges. It was generated by growing the original Arpanet with 40 nodes according to the random network model of [1]. As in Test 1, three instances were run, each one with all edges set to the same reliability. In this case the nodes s, t are represented as squares. Table 6 lists the scale of fines, split in three zones; again they were adjusted so that the expected value was similar for the different edge reliabilities. The heuristic PCCP was applied for $\mathcal{X}_1$ and it returned a solution built by one 12-pathset and two 5-cutsets. Heuristics PPPP and CCCC where applied to define the zones $\mathcal{X}_0$ and $\mathcal{X}_2$, returning respectively three pathsets and two cutsets. The parameter MAX_TIME was set again to 40 seconds. The results of the test are summarised in Table 7; each time reported for the proposed method include the 120 seconds spent generating the pathsets and cutsets. Note the significant relative efficiencies, in particular for rarer failures, obtained in this test: 21.92, 136.15 and 12,615.39 respectively for r_e equal to 0.90, 0.95 and 0.99.

7 Conclusions and future work

The proposed simulation method showed its capability to achieve significant variance reductions when applied on mesh-like networks. The precise conditions under which there is a reduction in the variance of the estimated parameter, with respect to the crude method, were shown in Lemma 4.1. The tests also

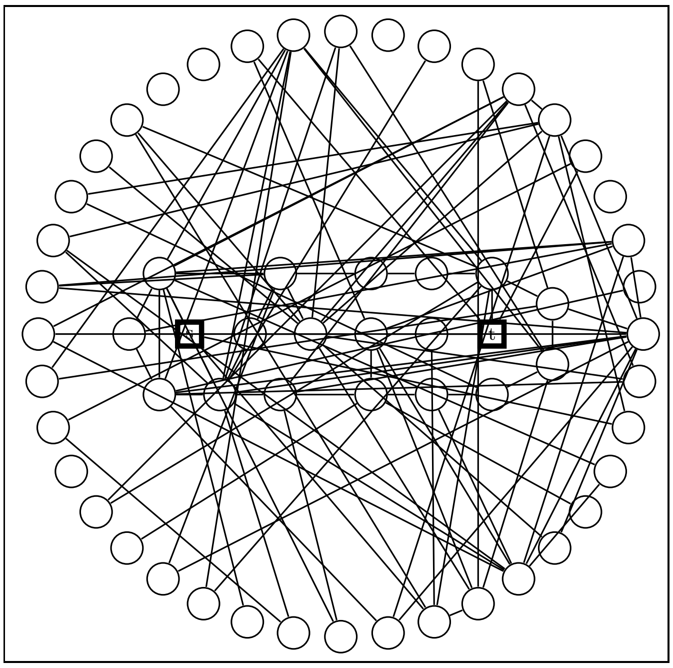

Figure 6: Network for Test 3 - Random extension of Arpanet

showed that the proposed heuristic was able to generate sets that exploited the mentioned variance-reduction potential at significant levels. Even considering the extra time required for running the heuristic and for sampling with the proposed plan, the efficiency gains are noteworthy, specially when the links become more reliable.

The heuristics hereby introduced, yet resulting in important efficiency gains when chained to the simulation in these simple versions, can be improved in several ways. The algorithm could adapt the amount of effort spent in searching higher cardinality sets of pathsets/cutsets according to statistics on the sets so far found or on the connectivity level of the terminal set. It could also alter the relative effort devoted to the generation of different sequences of pathsets and cutsets, reacting to the results so far obtained during execution. The reliability of each edge should be taken into account when choosing which one to add to the pathset or cutset under construction, particularly in networks where the reliabilities significantly differ. The time spent generating the sets MAX_TIME could also be initially set according to the sample size and the number of edges and regions (all of which determine at a large extent the time spent by the simulation). Moreover, it could be adjusted during the algorithm execution in light of the number and quality of the so far found sets.

References

[1] Réka Albert and Albert-László Barabási. Statistical mechanics of complex networks. *Rev. Mod. Phys.*, 74:47–97, Jan 2002.

[2] Z. I. Botev, P. L'Ecuyer, G. Rubino, R. Simard, and B. Tuffin. Static network reliability estimation via generalized splitting. *INFORMS Journal on Computing, to appear* -, 2012.

[3] H. Cancela, M. El Khadiri, and G. Rubino. *Rare event analysis by Monte Carlo techniques in static models. In Rare Event Simulation using Monte Carlo Methods, G. Rubino, B. Tuffin, Eds.*, chapter 7, pages 145–170. John Wiley & Sons, Chichester, 2009.

[4] H. Cancela, P. L'Ecuyer, M. Lee, G. Rubino, and B. Tuffin. *Analysis and improvements of path-based methods for Monte Carlo reliability evaluation of static models. In Simulation Methods for Reliability and Availability of Complex Systems, S. M. J. Faulin, A. A. Juan, and E. Ramirez-Marquez, Eds.*, pages 65 – 84. Springer-Verlag, Berlin, Germany, 2009.

[5] H. Cancela, P. L'Ecuyer, G. Rubino, and B. Tuffin. Combination of conditional monte carlo and approximate zero-variance importance sampling for network reliability estimation. In *Simulation Conference (WSC), Proceedings of the 2010 Winter*, pages 1263 –1274, dec. 2010.

[6] HÃ©ctor Cancela, Franco Robledo, Gerardo Rubino, and Pablo Sartor. Monte carlo estimation of diameter-constrained network reliability conditioned by pathsets and cutsets. *Computer Communications*, doi=10.1016/j.comcom.2012.08.010, 2012.

[7] Charles J. Colbourn. *The Combinatorics of Network Reliability.* Oxford University Press, Inc., New York, NY, USA, 1987.

[8] George S. Fishman. A Monte Carlo sampling plan for estimating network reliability. *Operations Research*, 34(4):581–594, july-august 1986.

[9] Ilya B. Gertsbakh and Yoseph Shpungin. *Models of Network Reliability: Analysis, Combinatorics, and Monte Carlo.* CRC Press, Inc., Boca Raton, FL, USA, 1st edition, 2009.

[10] Hiromitsu Kumamoto, Kazuo Tanaka, and Koichi Inoue. Efficient evaluation of system reliability by Monte Carlo method. *IEEE Transactions on Reliability*, R-26(5):311 –315, dec. 1977.

[11] Pierre L'Ecuyer, Gerardo Rubino, Samira Saggadi, and Bruno Tuffin. Approximate zero-variance importance sampling for static network reliability estimation. *IEEE Transactions on Reliability*, 60:590–604, 2011.

[12] L. Petingi and J. Rodriguez. Reliability of networks with delay constraints. In *Congressus Numerantium*, volume 152, pages 117–123, 2001.

[13] Louis Petingi. A diameter-constrained network reliability model to determine the probability that a communication network meets delay constraints. *WTOC*, 7:574–583, June 2008.

[14] G. Rubino and B. Tuffin. *Rare event simulation using Monte Carlo methods.* Wiley, 2009.

[15] Gerardo Rubino. *Network reliability evaluation. In State-of-the-art in performance modeling and simulation*, chapter 11. Gordon and Breach Books, 1996.

[16] Rico Zenklusen and Marco Laumanns. High-confidence estimation of small s-t reliabilities in directed acyclic networks. *Networks*, 57(4):376–388, 2011.

Contents

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-0803