Proceedings of the 2014 Winter Simulation Conference
A. Tolk, S. D. Diallo, I. O. Ryzhov, L. Yilmaz, S. Buckley, and J. A. Miller, eds.

RELIABILITY OF STOCHASTIC FLOW NETWORKS WITH CONTINUOUS LINK CAPACITIES

Zdravko I. Botev

School of Mathematics and Statistics
The University of New South Wales
Sydney, NSW 2052, Australia

Slava Vaisman

Department of Mathematics
The University of Queenland
Brisbane, QLD 4072, Australia

Reuven Y. Rubinstein

Faculty of Industrial Engineering and Management
Technion - Israel Institute of Technology
Haifa 32000, Israel

Pierre L'Ecuyer

DIRO, Université de Montreal
C.P. 6128, Succ. Centre-Ville
Montréal (Québec), H3C 3J7, Canada

ABSTRACT

We consider the problem of estimating the unreliability of a stochastic flow network, defined as the probability that the maximum flow value from a source node to a terminal node in a directed network with stochastic link capacities, is less than a specified demand level. The link capacities are assumed to be continuous random variables with a known joint distribution. We are interested in the situation where the unreliability is very small, in which case a crude Monte Carlo is not viable. We show how a Monte Carlo splitting algorithm can be adapted to handle this problem effectively.

1 INTRODUCTION

Let $\mathscr{G} = (\mathscr{V}, \mathscr{E})$ be a directed graph with $m = |\mathscr{E}|$ edges (or links) and a set of nodes or vertices $\mathscr{V}$, in which the i-th edge (link) has a random flow capacity $X_i \geqslant 0$. Denote $\mathbf{X} = (X_1, \ldots, X_m)$, the vector of flow capacities, and let f be the joint density of $\mathbf{X}$, over $[0,\infty)^m$. Let $\Psi(\mathbf{X})$ be the maximum flow that can be sent (say) from node 1 (the source) to node m (the sink), given the capacity vector $\mathbf{X}$ (Ford and Fulkerson 1962). We suppose that the network has to transport a given flow demand $d_{\text{net}} > 0$, and we are interested in estimating the *network unreliability* ℓ (one minus the reliability) defined as the probability that the network cannot satisfy the demand:

$$\ell = \ell(d_{\text{net}}) = \mathbb{P}(\Psi(\mathbf{X}) < d_{\text{net}}) = \int_{\Psi(\mathbf{x}) < d_{\text{net}}} f(\mathbf{x}) \mathrm{d}\mathbf{x}. \tag{1}$$

This problem occurs naturally in communication, transportation, and power distribution networks, for example, where link capacities are uncertain due to potential failures or degradations; see Gertsbakh and Shpungin (2012), Lin, Fiondella, and Chang (2013), and the references given there.

It is well known that the exact computation of the unreliability ℓ is an NP-hard problem (Ball 1986, Ball and Provan 1982, Colbourn 1987). This calls for alternatives such as Monte Carlo estimation techniques. The most straightforward (or crude) Monte Carlo method operates as follows. Generate $\mathbf{X}$ from density f and compute $Y = \mathbb{I}\{\Psi(\mathbf{X}) < d_{\text{net}}\}$, the indicator that the demand cannot be satisfied for the given $\mathbf{X}$. Repeat this n times independently, to obtain n realizations $Y_1, \ldots, Y_n$ of Y, and estimate ℓ by the average $\bar{Y}_n = (Y_1 + \cdots + Y_n)/n$. This estimator has variance $\mathrm{Var}[\bar{Y}_n] = \ell(1-\ell)/n$ and square *relative error* $\mathrm{RE}^2[\bar{Y}_n] = \mathrm{Var}[\bar{Y}_n]/\ell^2 = (1-\ell)/(\ell n)$. When ℓ is very small, which is typical, n must be very large to keep

the RE under control, and this eventually becomes impractical. Then, more refined Monte Carlo methods are needed.

Such Monte Carlo methods were proposed by Fishman (1989), followed by Fishman and Shaw (1989), Alexopoulos and Fishman (1993), Alexopoulos and Fishman (1992). This was later followed by the recursive algorithm of Bulteau and Khadiri (2002), based on an iterative graph decomposition method of Doulliez and Jamoulle (1972). All these algorithms operate under the assumptions that the random capacity of each network link must have a discrete distribution over a finite number of values, that this number is small, and that these random capacities are independent. In other words, the random vector $\mathbf{X}$ must take its value in the finite set $\Omega = \otimes_{i=1}^{m} \Omega_i$, where

$$\Omega_i = \{c_{i,1}, \ldots, c_{i,n_i}\}, \qquad 0 \leqslant c_{i,1} < c_{i,2} \cdots < c_{i,n_i} < \infty,$$

is the set of n_i possible values of X_i and $\mathbb{P}(\mathbf{X} = \mathbf{x}) = \prod_{i=1}^{m} \mathbb{P}(X_i = x_i)$ for all $\mathbf{x} \in \Omega$. As noted by Fishman (1989), the stochastic flow network model is realistic "provided one can justify the assumption of independent, discrete, integer-valued capacities". In reality, the independence assumption is made mainly to make the reliability computations tractable (Gertsbakh and Shpungin 2010).

These methods have limited efficiency when the network is large or the n_i are large. The method of Fishman (1989) requires finding minimal cut- and path- sets in the graph and then sampling from a discrete distribution on Ω with time complexity $\mathcal{O}(n_1 + \cdots + n_m)$. The algorithms in Alexopoulos and Fishman (1992), Alexopoulos and Fishman (1993), Bulteau and Khadiri (2002) are a significant improvement over the original method of (Fishman 1989), but they also all rely on the graph decomposition algorithm of Doulliez and Jamoulle (1972) (see Alexopoulos (1995) for the correction of a mistake in this algorithm). As a result, their worst-case time complexity is $\mathcal{O}(m \times n^3 \times (n_1 + \cdots n_m))$ (see Bulteau and Khadiri 2002, Lemmas 3 and 4, for example). This means in particular that approximating a continuous distribution by a discrete distribution over a large number of values would lead to a highly inefficient method. More recently, Gertsbakh and Shpungin (2012) have adapted the permutation Monte Carlo (PMC) method of Elperin, Gertsbakh, and Lomonosov (1991) to the stochastic flow network problem, but only under the assumption that each X_i can take only two possible values: zero and a fixed positive value (the link is failed or is operating). The idea of this method is to extend the static reliability model to a dynamic one, where each link fails at an exponential random time, generate only the *order* of the failures (a permutation), and compute the conditional probability that the demand is not satisfied given the permutation, as an estimator of the unreliability ℓ. This method is very effective when ℓ is very small and the network is not too large.

None of these proposed methods can handle continuous distributions for the link capacities, neither dependence across links. Here we take inspiration from what has been done recently for a different but closely related static reliability problem, where each link is either failed or operating, and we want to estimate the probability that two given nodes are not connected (the unreliability). This can be seen as a special case of the stochastic flow network problem considered here, where each capacity is 0 or 1, $d_{\text{net}} = 1$, and the two nodes that must be connected are labeled 1 and m. For this network connectivity problem, to estimate small unreliabilities, when the network is relatively small, the most effective method is usually the PMC method mentioned above and its refinement named the *turnip* method, proposed in Gertsbakh and Shpungin (2010). For very large networks, on the other hand, these methods break down and the most effective method we know is a generalized splitting (GS) algorithm introduced by Botev and Kroese (2012) and adapted to this problem in Botev, L'Ecuyer, Rubino, Simard, and Tuffin (2013). This is particularly true when the individual link unreliabilities $\mathbb{P}(X_i = 0)$ are not too small, but ℓ is very small because nodes 1 and m are connected by a large number of redundant paths. This GS approach also works for certain models with dependent links (Botev, L'Ecuyer, and Tuffin 2012). For more details, see Botev, L'Ecuyer, Rubino, Simard, and Tuffin (2013).

This motivated us to investigate if and how the GS method could be applied to the stochastic flow problem defined earlier. This is the purpose of this paper. The remainder is organized as follows. In Section 2, we state the GS algorithm in a general form. In Section 3, we provide a mathematical formulation

of the max-flow problem and recall an algorithm to solve this problem, the Edmonds-Karp algorithm. To adapt GS to our setting, we need an effective way to resample link capacities under certain conditional distributions, and appropriate methods that permit one to foresee and update quickly the maximum flow in the network when a link capacity is either increased or decreased. We address those key issues in Section 4. In Section 5, we provide a numerical example based on a popular test case originally given by Fishman (1989). Conclusions and suggestion for future research are given in Section 6.

2 GENERALIZED SPLITTING ALGORITHM

Intuitively, the GS method can be viewed as a way of estimating ℓ in (1) by constructing a sample of realizations of $\mathbf{X}$ approximately from it conditional distribution given that $\Psi(\mathbf{X}) < d_t$, using adaptive learning. We state a GS algorithm under the assumption that the capacities X_i are continuous random variables, with a density. Then, $\Psi(\mathbf{X})$ is also a continuous random variable and it can be used as the importance function in the GS procedure. We select an integer $s \geq 2$ called the *splitting factor* (usually $s = 2$ gives the best performance), and we introduce intermediate demand levels $\infty = d_0 > d_1 > d_2 > \cdots d_\tau = d_{\text{net}}$, for some integer $\tau > 0$, so that

$$\rho_t = \mathbb{P}(\Psi(\mathbf{X}) < d_t \,|\, \Psi(\mathbf{X}) < d_{t-1}) \approx 1/s, \quad t = 1, \ldots, \tau - 1, \tag{2}$$

and $\rho_\tau \leq 1/s$ (approximately). The intermediate demands d_t represent the levels in the GS algorithm, which gradually steers the vector $\mathbf{X}$ towards the rare-event regions of the sample space, where $\Psi(\mathbf{X})$ is small. At each level t we run a Markov chain $\{\mathbf{Y}_{t,j}, j \geqslant 0\}$ with a stationary density equal to the density of $\mathbf{X}$ conditional on $\Psi(\mathbf{X}) < d_t$. This density can be written as

$$f_t(\mathbf{x}) = f(\mathbf{x}) \frac{\mathbb{I}\{\Psi(\mathbf{x}) < d_t\}}{\mathbb{P}(\Psi(\mathbf{X}) < d_t)},$$

where $f_0 \equiv f$ is simply the unconditional density of the random capacities $\mathbf{X}$. We postpone the specification of the Markov chain $\{\mathbf{Y}_{t,j}, j \geqslant 0\}$ until Section 4. For the moment, we simply assume that this Markov chain is well-defined and can be easily simulated. We denote its transition density at level t by $\kappa(\cdot \,|\, \mathbf{X}_{t,j-1})$, where $\mathbf{X}_{t,j-1}$ is the current state, so $\mathbf{X}_{t,j} \sim \kappa(\mathbf{x} \,|\, \mathbf{X}_{t,j-1})$. The GS algorithm first samples from $f_0 \equiv f$ and then sequentially from $\kappa_1, \kappa_2, \ldots, \kappa_{\tau-1}$. It is stated as Algorithm 1. In all algorithms in this paper, the indentation delimits the scope of the **if**, **else**, and **for** statements.

This algorithm generates a single random variable W such that $\mathbb{E}[W] = u$; see (Botev and Kroese 2012). It will be invoked n times, independently, to produce n independent realizations of W, say $W_1, \ldots, W_n$, and ℓ is estimated by the sample average

$$\widehat{\ell} = \frac{1}{n} \sum_{i=1}^{n} W_i. \tag{3}$$

The squared relative error of $\widehat{\ell}$ is $\text{RE}^2[\widehat{\ell}] = \text{Var}[\widehat{\ell}]/\ell^2$, which can be estimated by

$$\widehat{\text{RE}}^2(\widehat{\ell}) = \frac{1}{n\widehat{\ell}^2} \sum_{i=1}^{n} (W_i - \widehat{\ell})^2.$$

A key issue is how do we choose the intermediate levels d_t to satisfy (2). These levels can be determined easily (approximately) using a pilot splitting algorithm, as described in Botev and Kroese (2012), Botev, L'Ecuyer, Rubino, Simard, and Tuffin (2013), and in the Appendix of Botev, L'Ecuyer, and Tuffin (2012). In the coming sections, we detail additional requirements: how to compute the maximum flow $\Psi(\mathbf{X})$, how to define the transition density κ_t, and how to sample from it efficiently.

Algorithm 1 : GS, returning W, an unbiased estimate of ℓ

Require: s, $d_0 > d_1 > d_2 > \cdots > d_\tau = d_{\text{net}}$
  Generate the random capacity vector $\mathbf{X}$ from its unconditional density f.
  if $\Psi(\mathbf{X}) < d_1$ **then**
    $\mathscr{X}_1 \leftarrow \{\mathbf{X}\}$
  else
    return $W \leftarrow 0$
  for $t = 2, \ldots, \tau$ **do**
    $\mathscr{X}_t \leftarrow \emptyset$ {set of capacity vectors that yield demand below d_t}
    for all $\mathbf{X}_0 \in \mathscr{X}_{t-1}$ **do**
      for $j = 1, \ldots, s$ **do**
        sample $\mathbf{X}_j$ from the density $\kappa_{t-1}(\cdot \mid \mathbf{X}_{j-1})$
        if $\Psi(\mathbf{X}_j) < d_t$ **then**
          add $\mathbf{X}_j$ to $\mathscr{X}_t$
  return $W \leftarrow |\mathscr{X}_\tau| / s^{\tau-1}$ as an unbiased unreliability estimate.

3 STOCHASTIC MAX-FLOW PROBLEM VIA MONTE CARLO

In the section we provide some background on the maximum flow (max-flow) problem in a network, and we recall the algorithm used in this paper to compute the maximum flow for given link capacities. The max-flow problem actually has applications in hundreds of areas. It appears as a building block (or subproblem) in various types of more complex optimization problems. Research on improving algorithms that solve this problem is also very rich and ongoing; see Ford and Fulkerson (1962), Edmonds and Karp (1972), Goldberg and Tarjan (1988), Goldberg, Tardos, and Tarjan (1990), Goldberg and Rao (1998), Kumar and Gupta (2003) for various algorithms to compute the maximum flow and Kelner, Lee, Orecchia, and Sidford (2013) for a recent account of the most novel methods. Here we opt for the older but simpler method of Edmonds and Karp (1972), whose time complexity is $\mathcal{O}(|\mathscr{V}| \times |\mathscr{E}|^2)$.

Let $\mathscr{C}_1, \ldots, \mathscr{C}_{c^*}$ be the list of all minimal cuts that separate nodes 1 and m. For a given vector of capacities $\mathbf{X} = \mathbf{x} = (x_1, \ldots, x_m)$, the flow capacity on cut $\mathscr{C}_1$ is $\sum_{i \in \mathscr{C}_j} x_i$, and the max-flow min-cut theorem (Ford and Fulkerson 1962) states that the maximum flow between nodes 1 and m in the network is equal to the flow capacity on the minimal cut having the smallest flow capacity:

$$\Psi(\mathbf{x}) = \min_{j=1,\ldots,c^*} \sum_{i \in \mathscr{C}_j} x_i. \tag{4}$$

For each link $(u,v) \in \mathscr{E}$, let $C_{u,v} \geqslant 0$ be its capacity, where $C_{u,v} = 0$ means that no link exists between nodes u and v, and let $F_{u,v}$ be the flow from node u to node v (a negative value represents a flow of $-F_{u,v}$ units from v to u). The max-flow problem can then be formulated as a linear programming problem, as follows (Ford and Fulkerson 1962):

$$\begin{aligned} &\text{maximize} \sum_{v \in \mathscr{V}} F_{v,m} \\ &\quad \text{subject to} \\ &\qquad F_{u,v} = -F_{v,u} \qquad \text{for all } (u,v) \in \mathscr{V} \times \mathscr{V} \\ &\qquad F_{u,v} \leqslant C_{u,v} \qquad \text{for all } (u,v) \in \mathscr{V} \times \mathscr{V} \\ &\qquad \sum_{v \in \mathscr{V}} F_{u,v} = 0 \qquad \text{for all } u \neq 1,\ v \neq m\,. \end{aligned} \tag{5}$$

The sum in the objective function represents the total flow landing in m. The first constraint states that a flow $F_{u,v}$ from u to v corresponds to a flow $-F_{u,v}$ from v to u. The second condition ensures that the flow

must respect the capacity constraint on each link. Finally, the third condition ensures that the net flow out of each node is zero, except for the source and sink.

Algorithm 2 describes a slightly modified version of the Edmonds-Karp algorithm to find the max-flow from 1 to m. The modification is that as soon as the algorithm finds a flow larger or equal to the target Ψ_{bound}, it stops. This early-exit option improves the efficiency significantly when we use the algorithm to update the max-flow in the GS method (see Section 4). If $\Psi_{\text{bound}} = \infty$, the algorithm always delivers the maximum flow. The algorithm searches for a path along which the flow from 1 to m can be increased, and increases that flow, until no such path can be found or the target flow Ψ_{bound} is reached.

Algorithm 2 : Maximum flow computation via Edmonds-Karp.

Require: the capacity matrix C and the target flow Ψ_{bound} (a bound on the max-flow), possibly ∞

Set the flow matrix to the zero matrix $F \leftarrow 0$

loop

Compute the current net flow $\Psi \leftarrow \sum_{v=1}^{m} F_{1,v}$

if $\Psi = \Psi_{\text{bound}}$ **then**

return the max-flow (or its upper bound) $\Psi(\mathbf{x}) = \Psi$

else

Compute the residual capacity matrix $R \leftarrow C - F$.

if a shortest path $\mathscr{P}$ can be found between nodes 1 and m in the network with capacity matrix R **then**

Compute the max-flow along path $\mathscr{P}$, given by the capacity of the bottleneck link:

$$c_{\mathscr{P}} \leftarrow \min_{(u,v)\in\mathscr{P}} R_{u,v}$$

Update the flow matrix F, making sure that the overall network flow does not exceed Ψ_{bound}:

$$F_{u,v} \leftarrow F_{u,v} + \min\{c_{\mathscr{P}}, \Psi_{\text{bound}} - \Psi\}, \quad \forall (u,v) \in \mathscr{P}$$

Update the reciprocal flow:

$$F_{v,u} \leftarrow -F_{u,v}, \quad \forall (u,v) \in \mathscr{P}$$

else

return the maximum (or upper bounded) network flow $\Psi(\mathbf{x}) = \Psi$

4 RESAMPLING AND UPDATING THE LINK CAPACITIES IN GS

Resampling Via Gibbs Sampling. We now explain how we sample from $\kappa_{t-1}(\cdot \mid \mathbf{X}_{j-1})$ in the GS Algorithm 1. For each t, we define κ_t as the transition density of a systematic Gibbs sampler, which resamples the capacity X_i of each link i, conditional on $\Psi(\mathbf{X}) < d_t$ given the current capacities of all the other links, in the order $i = 1, 2, \ldots, m$. The conditional resampling of link i is stated in Algorithm 3. The idea is simple: Given the current X_i and current flow Ψ, we first check what would happen if we increase the capacity X_i to $X_i^* = X_i + d_t - \Psi$. If this increases the flow to d_t, this means that link i is a bottleneck link and remains so even after this increase, and therefore X_i must be resampled conditionally on $X_i < X_i^*$. Otherwise, there is no constraint on X_i because the bottleneck is elsewhere, so it can be resampled from its distribution over $[0, \infty)$. In both cases, the resampling is done conditionally on the other coordinates of $\mathbf{X}$. This last conditioning matters when the X_i are not independent under the original density f.

Algorithm 3 Resampling of capacity X_i of link i conditional on $\Psi(\mathbf{X}) < d_t$.

Require: Current vector of capacities $\mathbf{X}$, with corresponding max-flow $\Psi = \Psi(\mathbf{X})$.
Set the capacity of link i temporarily to $X_i^* = X_i + d_t - \Psi$ and compute the max-flow Ψ^*.
if $\Psi^* < d_t$ **then**
sample the new capacity X_i^{new} from its original pdf $f(x_i \,|\, X_1, \ldots, X_{i-1}, X_{i+1}, \ldots, X_m)$
else
sample the new capacity X_i^{new} from $f(x_i \,|\, X_1, \ldots, X_{i-1}, X_{i+1}, \ldots, X_m)$ conditional on $X_i^{\text{new}} < X_i^*$.

Dynamic Updating of the Max-Flow. To implement the Gibbs sampler efficiently, we need an efficient method to verify the condition $\Psi^* < d_t$ in Algorithm 3 and to update the maximum flow of the network when the capacity of one link is changed. The naive approach that recomputes the max-flow from scratch via Algorithm 1 after each X_i is resampled has time complexity $\mathcal{O}(|\mathscr{V}| \times |\mathscr{E}|^3)$ and is much too inefficient. Instead, we use a dynamic updating scheme of the max-flow in the Edmonds-Karp algorithm. A similar updating scheme could also be defined for other max-flow algorithms.

To describe our dynamic updating scheme, we assume that the current flow and current capacity of all links are in matrices F and C, respectively, as in Section 3. We do this only for the notation, it does not imply that these are actually stored in arrays in the implementation. Suppose the current maximum flow is $\Psi = \sum_{v=2}^{m} F_{1,v}$, that we consider a change of the capacity of link (u,v) from its current value of $C_{u,v}$ to the new value of $c(u,v)$, and we want to update the max-flow to its new value Ψ_{new}. We distinguish two situations: the case where $C_{u,v}$ is increased and the case where it is decreased.

Increased link capacity. If $c(u,v) > C_{u,v}$, the max-flow can either increase or remain the same, depending on whether (u,v) is a bottleneck link or not. If $F_{u,v} < C_{u,v}$, the link is underutilized, so increasing its capacity will not change the max-flow, it will only increase the unused capacity of that link, from $C_{u,v} - F_{u,v}$ to $c(u,v) - F_{u,v}$. If $F_{u,v} = C_{u,v}$, the link is fully utilized (it is a bottleneck), so increasing its capacity may increase the max-flow. After updating $C_{u,v}$ to $c(u,v)$, we search for paths along which the flow can be increased, in the residual network whose link capacities are given by the residual capacity matrix $R = C - F$. If no such paths exist, then F and the maximum flow remain the same (this means that at least one other link is also a bottleneck). If some (one or more) augmenting paths are found that give rise to a nonzero flow matrix F_{aug} over the residual graph, then the updated flow is $F_{\text{new}} = F + F_{\text{aug}}$ and the maximum flow is now $\Psi_{\text{new}} = \Psi + \Psi_{\text{aug}}$, where Ψ_{aug} is the maximum flow carried by the augmenting paths.

Reduced link capacity. Suppose $c(u,v) < C_{u,v}$, i.e., the capacity of link (u,v) is reduced. If $F_{u,v} \leqslant c(u,v)$, then the new capacity is still sufficient to handle the current flow on that link, and therefore there is no change on the flow. If $F_{u,v} > c(u,v)$, then there is an amount of flow of $\Delta = F_{u,v} - c(u,v)$ that can no longer pass thought this link. The max-flow Ψ may be reduced by Δ, unless this extra flow can be re-routed (in part or in totality), in which case the flow reduction can be less than Δ. To find paths along which this flow can be re-routed, we start by replacing $F_{u,v}$ by $c(u,v)$ and $F_{v,u}$ by $-c(u,v)$ in the matrix F, we put $C_{u,v} = c(u,v)$, and we compute the matrix $R = C - F$ of residual capacities. Then we apply the max-flow algorithm to transfer the maximum flow (but no more than Δ) from node u to node v in the network with capacity matrix R. Let F_{reroute} be the flow matrix found by this subproblem and $\Psi_{\text{reroute}} \leq \Delta$ be the corresponding max-flow transferred from u to v. The new (updated) max-flow in the network is now $\Psi_{\text{new}} = \Psi - \Delta + \Psi_{\text{reroute}}$. Updating the flow matrix F is a bit more involved. Note that after setting $F_{u,v} = c(u,v)$, the flow balance equation (5) no longer holds at nodes u and v, because there is an excess flow of $\Delta - \Psi_{\text{reroute}}$ at node u and a deficiency of $\Delta - \Psi_{\text{reroute}}$ at node v. To rebalance the flow matrix, we push the excess flow $\Delta - \Psi_{\text{reroute}}$ back from node u to 1 (unless $u = 1$ already) along the shortest augmenting paths that we can find. We also push flow from the sink node m to node v in the same way.

The procedure just described is stated in Algorithm 4. We use it as a subroutine in the systematic Gibbs sampler to update the flow matrix F and the corresponding max-flow Ψ.

Algorithm 4 Max-flow updating after changing the capacity of link $i = (u, v)$.

Require: current flow matrix F, capacity matrix C, new link capacity $c(u,v)$, and max-flow Ψ

if $c(u,v) > C_{u,v}$ **then**
  if $F_{u,v} < C_{u,v}$ **then**
    return $F_{\text{new}} \leftarrow F$ and $\Psi_{\text{new}} \leftarrow \Psi$
  else
    Set $C_{u,v} \leftarrow c(u,v)$ and use Algorithm 2 with $\Psi_{\text{bound}} = \infty$ to compute the max-flow F_{aug} and Ψ_{aug} from node 1 to node m, in the network with capacity matrix $R = C - F$
    return $F_{\text{new}} \leftarrow F + F_{\text{aug}}$ and $\Psi_{\text{new}} \leftarrow \Psi + \Psi_{\text{aug}}$
else
  if $F_{u,v} \leqslant c(u,v)$ **then**
    return $F_{\text{new}} \leftarrow F$ and $\Psi_{\text{new}} \leftarrow \Psi$
  else
    Set $F_{u,v} \leftarrow c(u,v)$, $F_{v,u} \leftarrow -c(u,v)$, $C_{u,v} \leftarrow c(u,v)$, and use Algorithm 2 with $\Psi_{\text{bound}} = \Delta$ to compute the maximum flow F_{reroute} and Ψ_{reroute} from u to v, in the network with capacity matrix $R = C - F$
    if $u \neq 1$ **then**
      Use Algorithm 2 with $\Psi_{\text{bound}} = \Delta - \Psi_{\text{reroute}}$ to compute the flow matrix F^* giving the max-flow from u to 1 in the network with capacity matrix R
      $F_{\text{new}} \leftarrow F_{\text{new}} + F^*$
    if $v \neq m$ **then**
      Use Algorithm 2 with $\Psi_{\text{bound}} = \Delta - \Psi_{\text{reroute}}$ to compute the flow matrix F^* giving the max-flow from m to v in the network with capacity matrix R
      $F_{\text{new}} \leftarrow F_{\text{new}} + F^*$
    return $\Psi_{\text{new}} \leftarrow \Psi - \Delta + \Psi_{\text{reroute}}$ and $F_{\text{new}} \leftarrow F_{\text{new}} + F_{\text{reroute}}$

5 NUMERICAL ILLUSTRATION

Consider the network in Figure 1, with $|\mathscr{V}| = 10$ nodes and $m = 25$ links, labeled from 1 to 25. This example has been widely used as a test problem (Fishman 1989, Fishman and Shaw 1989, Alexopoulos and Fishman 1993, Alexopoulos and Fishman 1992). Suppose that the link capacities $X_1, \ldots, X_m$ are independent and have the uniform distribution over $(0, 1000)$. We want to estimate the probability ℓ that the maximum flow from node 1 to node 10 is less than d_{net}. We use the GS estimator (3), with splitting factor $s = 2$ and sample size $n = 5000$. Note that the expected total simulation effort is approximately proportional to $n \times s \times \tau$, where $\tau \approx \lceil -\ln_2(\widehat{\ell}) \rceil$ is the number of levels in the GS procedure.

Table 1 shows the estimated unreliability $\widehat{\ell}$ obtained by this procedure, as well as the estimated relative error $\widehat{\text{RE}}(\widehat{\ell})$, the expected simulation effort $n \times s \times \tau$, and the efficiency gain over crude Monte Carlo, for different values of the network demand d_{net}. The efficiency gain is defined as the work-normalized variance of the crude Monte Carlo estimator divided by that of the GS estimator. Equivalently, it is defined as the simulation effort required by the crude estimator divided by the simulation effort required by GS to achieve the same variance (or same relative error).

We see from the table that GS yields significant gains over the naive estimator when the network is highly reliable, and this gain increases when the unreliability ℓ decreases.

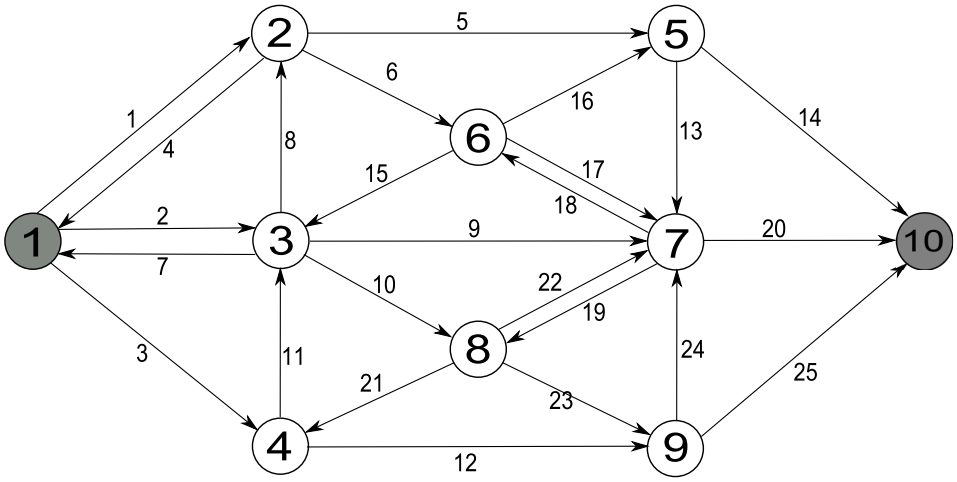

Figure 1: A network with 25 links, taken from Fishman (1989)

Table 1: Unreliability estimates for the network in Figure 1, for different values of d_{net}.

d_{net}	$\widehat{\ell}$	$\widehat{\text{RE}}(\widehat{\ell})$	simulation effort	efficiency gain
250	0.0061	2.8%	7×10^4	3
200	0.0029	2.8%	8×10^4	5
150	0.0011	2.8%	1.0×10^5	10
100	0.00037	3.3%	1.1×10^5	20
50	4.3×10^{-5}	4.8%	1.5×10^5	70
40	2.1×10^{-5}	4.8%	1.6×10^5	100
30	9.2×10^{-6}	3.6%	1.7×10^5	500
20	2.7×10^{-6}	4.4%	1.9×10^5	1000
10	3.4×10^{-7}	5.4%	2.1×10^5	4500

6 CONCLUSION

We have shown how to use a Monte Carlo splitting technique to estimate the probability that the maximum flow in a stochastic network fails to meet a predetermined demand, when this probability is small. Our GS procedure works well in the case where the link capacities have a continuous distribution, a situation not covered by previous methods.

There are a number of different possibilities for future exploration. First, we need further numerical investigation, on larger examples, and on examples where the link capacities are dependent. Dependent link capacities can be handled provided that we can perform the conditional resampling in Algorithm 3. One type of situation where this can be achieved is if $\mathbf{X}$ has a multivariate normal distribution, or a distribution specified by a normal copula. Second, one can explore more efficient dynamic updating algorithms to speed up the Gibbs sampler based on recently proposed novel max-flow algorithms. Finally, all methods considered so far have assumed that the network demand was fixed in advance. However, a more realistic model might consider a random demand d_{net}.

ACKNOWLEDGMENTS

Zdravko Botev has been supported by the *Australian Research Council Discovery Early Career Researcher Award* RG133085 and the *Early Career Researcher Grant* of the School of Mathematics and Statistics at the University of New South Wales (UNSW), Sydney, Australia. Pierre L'Ecuyer received support from an NSERC-Canada Discovery Grant, a Canada Research Chair, and an Inria International Chair. This paper was written during his visit at UNSW, supported by the *Faculty of Science Visiting Researcher Award*.

REFERENCES

Alexopoulos, C. 1995. "A note on state-space decomposition methods for analyzing stochastic flow networks". *IEEE Transactions on Reliability* 44 (2): 354–357.

Alexopoulos, C., and G. S. Fishman. 1992. "Capacity Expansion in Stochastic Flow Networks". *Probability in the Engineering and Informational Sciences* 6:99–118.

Alexopoulos, C., and G. S. Fishman. 1993. "Sensitivity Analysis in Stochastic Flow Networks Using the Monte Carlo method". *Networks* 23:605–621.

Ball, M. O. 1986. "Computational complexity of network reliability analysis: An Overview". *IEEE Transactions on Reliability* 35 (3): 230–239.

Ball, M. O., and J. S. Provan. 1982. "Bounds on the Reliability Polynomial for Shellable Independence Systems". *SIAM Journal on Algebraic and Discrete Methods* 3:166–181.

Botev, Z. I., and D. P. Kroese. 2012. "Efficient Monte Carlo simulation via the generalized splitting method". *Statistics and Computing* 22 (1): 1–16.

Botev, Z. I., P. L'Ecuyer, G. Rubino, R. Simard, and B. Tuffin. 2013. "Static network reliability estimation via generalized splitting". *INFORMS Journal on Computing* 25 (1): 56–71.

Botev, Z. I., P. L'Ecuyer, and B. Tuffin. 2012. "Dependent failures in highly reliable static networks". In *Proceedings of the 2012 Winter Simulation Conference*, 430–441. IEEE Press.

Bulteau, S., and M. E. Khadiri. 2002. "A New Importance Sampling Monte Carlo Method for a Flow Network Reliability Problem". *Naval Research Logistics* 49 (2): 204–228.

Colbourn, C. J. 1987. *The Combinatorics of Network Reliability*. New York: Oxford University Press.

Doulliez, P., and E. Jamoulle. 1972. "Transportation networks with random arc capacities". *R.A.I.R.O.*:45–59.

Edmonds, J., and R. M. Karp. 1972. "Theoretical improvements in algorithmic efficiency for network flow problems". *Journal of the ACM* 19 (2): 248–264.

Elperin, T., I. B. Gertsbakh, and M. Lomonosov. 1991. "Estimation of Network Reliability Using Graph Evolution Models". *IEEE Transactions on Reliability* 40 (5): 572–581.

Fishman, G. S. 1989. "Monte Carlo Estimation of the Maximal Flow Distribution with Discrete Stochastic Arc Capacity Levels". *Naval Research Logistics* 36 (4): 829–849.

Fishman, G. S., and T.-Y. D. Shaw. 1989. "Evaluating Reliability of Stochastic Flow Networks". *Probability in the Engineering and Informational Sciences* 3:493–509.

Ford, L. R., and D. R. Fulkerson. 1962. *Flows in Networks*. New Jersey: Princeton University Press.

Gertsbakh, I., and Y. Shpungin. 2012. "Spectral Approach to Reliability Evaluation of Flow Networks". In *Proceedings of the European Modeling and Simulation Symposium*, 68–73.

Gertsbakh, I. B., and Y. Shpungin. 2010. *Models of Network Reliability: Analysis, Combinatorics, and Monte Carlo*. Boca Raton, FL: CRC Press.

Goldberg, A. V., and S. Rao. 1998. "Beyond the flow decomposition barrier". *Journal of the ACM* 45 (5): 783–797.

Goldberg, A. V., E. Tardos, and R. E. Tarjan. 1990. "Network Flow Algorithms". In *Algorithms and Combinatorics: Paths, Rows, and VLSI-Layout*, Volume 9, 101–164. Berlin: Springer-Verlag.

Goldberg, A. V., and R. E. Tarjan. 1988. "A New Approach to the Maximum-Flow Problem". *Journal of the ACM* 35 (4): 921–940.

Kelner, J., Y. Lee, L. Orecchia, and A. Sidford. 2013. *An Almost-Linear-Time Algorithm for Approximate Max Flow in Undirected Graphs, and its Multicommodity Generalizations*, Chapter 16, 217–226.

Kumar, S., and P. Gupta. 2003. "An Incremental Algorithm for the Maximum Flow Problem". *Journal of Mathematical Modelling and Algorithms* 2:1–16.

Lin, Y.-K., L. Fiondella, and P.-C. Chang. 2013. "Quantifying the impact of correlated failures on system reliability by a simulation approach". *Reliability Engineering and System Safety* 109:32–40.

AUTHOR BIOGRAPHIES

ZDRAVKO I. BOTEV is a Lecturer at the School of Mathematics and Statistics at the University of New South Wales in Sydney, Australia. He obtained his Ph.D. in Mathematics from The University of Queensland, Australia, in 2010. His research interests include splitting and adaptive importance sampling methods for rare-event simulation. He has written jointly with D. P. Kroese and T. Taimre a *Handbook of Monte Carlo Methods* published by John Wiley & Sons in 2011.

SLAVA VAISMAN is a Postdoctoral Research Fellow at the University of Queensland in Brisbane, Australia. He obtained his PhD in Operations Research from the Technion, Haifa, Isreal in 2013. His research interests include rare event simulation, randomized algorithms, and on-line planning. He has written jointly with Reuven Rubinstein and Ad Ridder a monograph on *Fast Sequential Monte Carlo Methods for Counting and Optimization* published by Wiley in 2013.

REUVEN R. RUBINSTEIN was Professor at the Faculty of Industrial Engineering and Management of the Technion, Israel. Professor Rubinstein passed away in December 2012. He was a member of several societies including the Operations Research Society of Israel and the American Operations Research Society. Having written numerous books and articles throughout his long career, Rubinstein is widely-known as a pioneer of the score function (SF) method in simulation and the cross-entropy (CE) method for combinatorial optimization. His research was recognised with several awards, including the 2010 Lifetime Professional Achievement Award from the INFORMS Simulation Society, and the 2011 Lifetime Achievement Award from ORSIS.

PIERRE L'ECUYER is Professor in the Département d'Informatique et de Recherche Opérationnelle, at the Université de Montréal, Canada. He holds the Canada Research Chair in Stochastic Simulation and Optimization and an Inria International Chair in Rennes, France. He is a member of the CIRRELT and GERAD research centers. His main research interests are random number generation, quasi-Monte Carlo methods, efficiency improvement via variance reduction, sensitivity analysis and optimization of discrete-event stochastic systems, and discrete-event simulation in general. He has served as Editor-in-Chief for *ACM Transactions on Modeling and Computer Simulation* from 2010 to 2013. He is currently Associate Editor for *ACM Transactions on Mathematical Software*, *Statistics and Computing*, *International Transactions in Operational Research*, and *Cryptography and Communications*. More information can be found on his web page: http://www.iro.umontreal.ca/~lecuyer.