Modeling and Estimating Small Unreliabilities for Static Networks with Dependent Components

Pierre L'Ecuyer
Université de Montréal, Canada, and Inria–Rennes, France

Zdravko Botev, New South Wales University, Australia
Richard Simard, Université de Montréal, Canada
Bruno Tuffin, Inria–Rennes, France

SNA-MC, Paris, October 2013

A static network reliability problem

A system has m components, in state 0 (failed) or 1 (operating).
System state: $\mathbf{X} = (X_1, \ldots, X_m)^t$.
Complementary structure function: $\Phi : \{0,1\}^m \to \{0,1\}$.
System failed iff $\Phi(\mathbf{X}) = 1$. Unreliability: $u = \mathbb{P}[\Phi(\mathbf{X}) = 1]$.

A static network reliability problem

A system has m components, in state 0 (failed) or 1 (operating).
System state: $\mathbf{X} = (X_1, \ldots, X_m)^t$.
Complementary structure function: $\Phi : \{0,1\}^m \to \{0,1\}$.
System failed iff $\Phi(\mathbf{X}) = 1$. Unreliability: $u = \mathbb{P}[\Phi(\mathbf{X}) = 1]$.

If we know $\mathbb{P}[\mathbf{X} = \mathbf{x}]$ for all $\mathbf{x} \in \{0,1\}^m$, in theory we can compute

$$u = \sum_{\mathbf{x} \in \mathcal{D} = \{\mathbf{X} : \Phi(\mathbf{X}) = 1\}} \mathbb{P}[\mathbf{X} = \mathbf{x}].$$

But the cost of enumerating $\mathcal{D}$ is generally exponential in m.

A static network reliability problem

A system has m components, in state 0 (failed) or 1 (operating).
System state: $\mathbf{X} = (X_1, \ldots, X_m)^t$.
Complementary structure function: $\Phi : \{0,1\}^m \to \{0,1\}$.
System failed iff $\Phi(\mathbf{X}) = 1$. Unreliability: $u = \mathbb{P}[\Phi(\mathbf{X}) = 1]$.

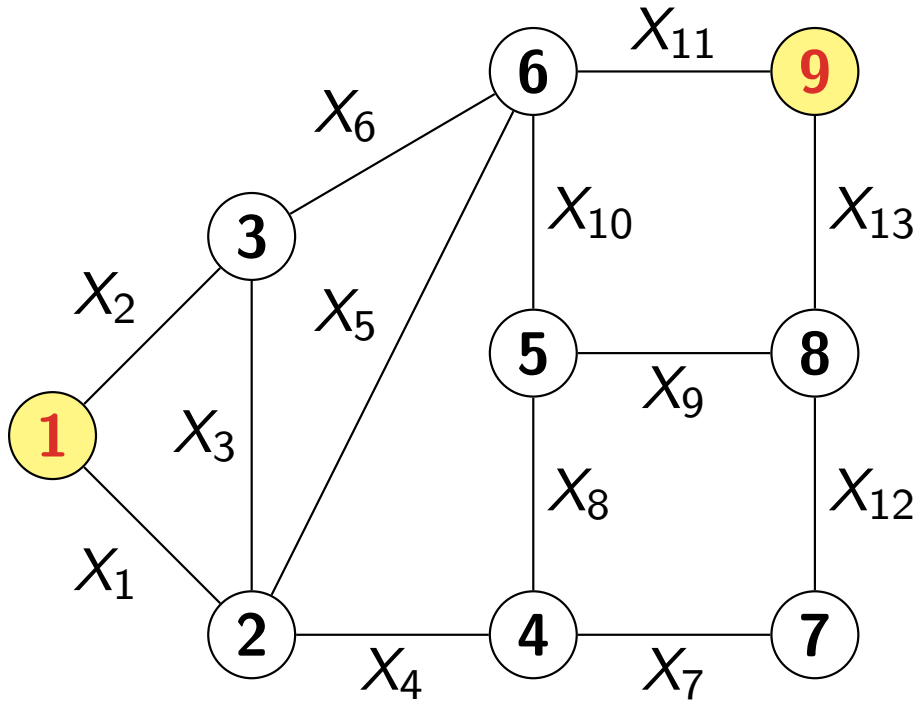
If we know $\mathbb{P}[\mathbf{X} = \mathbf{x}]$ for all $\mathbf{x} \in \{0,1\}^m$, in theory we can compute

$$u = \sum_{\mathbf{x} \in \mathcal{D} = \{\mathbf{X} : \Phi(\mathbf{X}) = 1\}} \mathbb{P}[\mathbf{X} = \mathbf{x}].$$

But the cost of enumerating $\mathcal{D}$ is generally exponential in m.

Monte Carlo: Generate n i.i.d. realizations of $\mathbf{X}$, say $\mathbf{X}_1, \ldots, \mathbf{X}_n$, compute $W_i = \Phi(\mathbf{X}_i)$ for each i, and estimate u by $\bar{W}_n = (W_1 + \cdots + W_n)/n$.

Suppose Φ is defined via a graph. Link i "works" iff $X_i = 1$.
The system works if all the nodes in a given set $\mathcal{V}_0$ are connected.



The X_i's can be independent or not.
Given the X_i's, $W = \Phi(\mathbf{X})$ is easy to evaluate by graph algorithms (e.g., minimal spanning tree).

But the failure probability u is often very close to 0 (failure is a rare event). For example, if $u = 10^{-10}$, the system will fail only once per 10 billion runs on average.

But the failure probability u is often very close to 0 (failure is a rare event). For example, if $u = 10^{-10}$, the system will fail only once per 10 billion runs on average.

In fact, $W = \Phi(\mathbf{X})$ is Bernoulli with $\mathbb{E}[W] = u$, $\mathrm{Var}[W] = u(1-u)$, and

$$\mathrm{MSE}[\bar{W}_n] \overset{\text{here}}{=} \mathrm{Var}[\bar{W}_n] = \frac{u(1-u)}{n} \approx \frac{u}{n}.$$

We want at least to beat the trivial estimator $Y = 0$, for which $\mathrm{MSE}[Y] = \mathrm{bias}^2[Y] = u^2$.

But the failure probability u is often very close to 0 (failure is a rare event). For example, if $u = 10^{-10}$, the system will fail only once per 10 billion runs on average.

In fact, $W = \Phi(\mathbf{X})$ is Bernoulli with $\mathbb{E}[W] = u$, $\mathrm{Var}[W] = u(1-u)$, and

$$\mathrm{MSE}[\bar{W}_n] \overset{\text{here}}{=} \mathrm{Var}[\bar{W}_n] = \frac{u(1-u)}{n} \approx \frac{u}{n}.$$

We want at least to beat the trivial estimator $Y = 0$, for which $\mathrm{MSE}[Y] = \mathrm{bias}^2[Y] = u^2$.

When u is small, a relevant quality measure is the relative error

$$\mathrm{RE}[\bar{W}_n] \overset{\text{def}}{=} \frac{\sqrt{\mathrm{MSE}[\bar{W}_n]}}{u} \overset{\text{here}}{=} \frac{\sqrt{1-u}}{\sqrt{nu}} \to \infty \quad \text{when } u \to 0.$$

For example, if $u \approx 10^{-10}$, we need $n \approx 10^{12}$ to have $\mathrm{RE}[\bar{W}_n] \leq 10\%$. We need much more efficient methods than crude MC!

A framework for asymptotic analysis

We parameterize the distribution of $\mathbf{X}$ by some rarity parameter ϵ in a way that $u = u(\epsilon) = \mathbb{P}[W(\epsilon) = 1] \to 0$ when $\epsilon \to 0$.

For example, one can have

$$\mathbb{P}[X_i = 0] = q_i = a_i \epsilon^{b_i} + o(\epsilon^{b_i})$$

for each link i, with independent X_i's.

We study the asymptotic behavior when $\epsilon \to 0$ to understand what happens when ϵ is very small.

With standard MC, $\mathrm{RE}[W(\epsilon)] \to \infty$ when $\epsilon \to 0$.

Robustness of estimator $W(\epsilon)$ in a rare-event setting

An estimator $W(\epsilon)$ with mean $\mu_0(\epsilon)$ has bounded relative variance, or bounded relative error (BRE) if

$$\lim_{\epsilon \to 0} \frac{\mathrm{Var}[W(\epsilon)]}{\mu_0^2(\epsilon)} < \infty.$$

Then the relative size of a confidence interval on the mean remains bounded when $\epsilon \to 0$.

Robustness of estimator $W(\epsilon)$ in a rare-event setting

An estimator $W(\epsilon)$ with mean $\mu_0(\epsilon)$ has bounded relative variance, or bounded relative error (BRE) if

$$\lim_{\epsilon \to 0} \frac{\mathrm{Var}[W(\epsilon)]}{\mu_0^2(\epsilon)} < \infty.$$

Then the relative size of a confidence interval on the mean remains bounded when $\epsilon \to 0$. Can we do better?

$W(\epsilon)$ has vanishing relative variance, or relative error (VRE), if

$$\limsup_{\epsilon \to 0} \frac{\sigma(\epsilon)}{\mu_0(\epsilon)} = 0.$$

Then the rare event difficulty is reversed! May seem unachievable at first sight, but does happen.

Challenge in rare-event simul.: build estimators with these properties.

Conditional MC with auxiliary variables

[Elperin, Gertsbach, Lomonosov 1974, 1991, 1992, etc.]
Special case: the X_i's are independent with $\mathbb{P}[X_i = 0] = q_i$.

Conceptually, suppose each link i is initially failed and gets repaired at time $Y_i \sim \text{Expon}(\mu_i)$ where $\mu_i = -\ln(q_i)$. Then $\mathbb{P}[Y_i > 1] = \mathbb{P}[X_i = 0] = q_i$.
Let $\mathbf{Y} = (Y_1, \ldots, Y_m)$ and π the permutation s.t. $Y_{\pi(1)} < \cdots < Y_{\pi(m)}$.

Conditional on π, we can forget the Y_i's, add the (non-redundant) links one by one until the graph is operational, say at step C.
Data structure: forest of spanning trees. Adding a link may merge two trees.

Conditional MC with auxiliary variables

[Elperin, Gertsbach, Lomonosov 1974, 1991, 1992, etc.]
Special case: the X_i's are independent with $\mathbb{P}[X_i = 0] = q_i$.

Conceptually, suppose each link i is initially failed and gets repaired at time $Y_i \sim \text{Expon}(\mu_i)$ where $\mu_i = -\ln(q_i)$. Then $\mathbb{P}[Y_i > 1] = \mathbb{P}[X_i = 0] = q_i$.
Let $\mathbf{Y} = (Y_1, \ldots, Y_m)$ and π the permutation s.t. $Y_{\pi(1)} < \cdots < Y_{\pi(m)}$.

Conditional on π, we can forget the Y_i's, add the (non-redundant) links one by one until the graph is operational, say at step C.
Data structure: forest of spanning trees. Adding a link may merge two trees.

Permutation Monte Carlo (PMC) estimator: conditional probability that the total time for these repairs is larger than 1:

$$\mathbb{P}\left[A_1 + \cdots + A_c > 1 \mid \pi,\, C = c\right].$$

At step j, the time A_j to next repair is exponential with rate Λ_j, the sum of repair rates of all links not yet repaired. Sum is an hypoexponential.
Theorem [Gertsback and Shpungin 2010]. Gives BRE when the $q_j \to 0$.

Conditional MC with auxiliary variables

[Elperin, Gertsbach, Lomonosov 1974, 1991, 1992, etc.]
Special case: the X_i's are independent with $\mathbb{P}[X_i = 0] = q_i$.

Conceptually, suppose each link i is initially failed and gets repaired at time $Y_i \sim \text{Expon}(\mu_i)$ where $\mu_i = -\ln(q_i)$. Then $\mathbb{P}[Y_i > 1] = \mathbb{P}[X_i = 0] = q_i$.
Let $\mathbf{Y} = (Y_1, \ldots, Y_m)$ and π the permutation s.t. $Y_{\pi(1)} < \cdots < Y_{\pi(m)}$.

Conditional on π, we can forget the Y_i's, add the (non-redundant) links one by one until the graph is operational, say at step C.
Data structure: forest of spanning trees. Adding a link may merge two trees.

Permutation Monte Carlo (PMC) estimator: conditional probability that the total time for these repairs is larger than 1:

$$\mathbb{P}\left[A_1 + \cdots + A_c > 1 \mid \pi,\, C = c\right].$$

At step j, the time A_j to next repair is exponential with rate Λ_j, the sum of repair rates of all links not yet repaired. Sum is an hypoexponential.
Theorem [Gertsback and Shpungin 2010]. Gives BRE when the $q_j \to 0$.

Improvement: turnip; at each step, discard redundant unrepaired links.

$$\mathbb{P}\left[A_1 + \cdots + A_c > t \mid \pi,\, C = c\right] = \mathbf{e}_1 e^{\mathbf{D}t}\mathbf{1} = \mathbf{e}_1 \sum_{k=0}^{\infty} \frac{\mathbf{D}^k\, t^k}{k!}\mathbf{1},$$

where

$$\mathbf{D} = \begin{pmatrix} -\Lambda_1 & \Lambda_1 & 0 & \dots & 0 \\ 0 & -\Lambda_2 & \Lambda_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & -\Lambda_{c-1} & \Lambda_{c-1} \\ 0 & \dots & 0 & 0 & -\Lambda_c \end{pmatrix}.$$

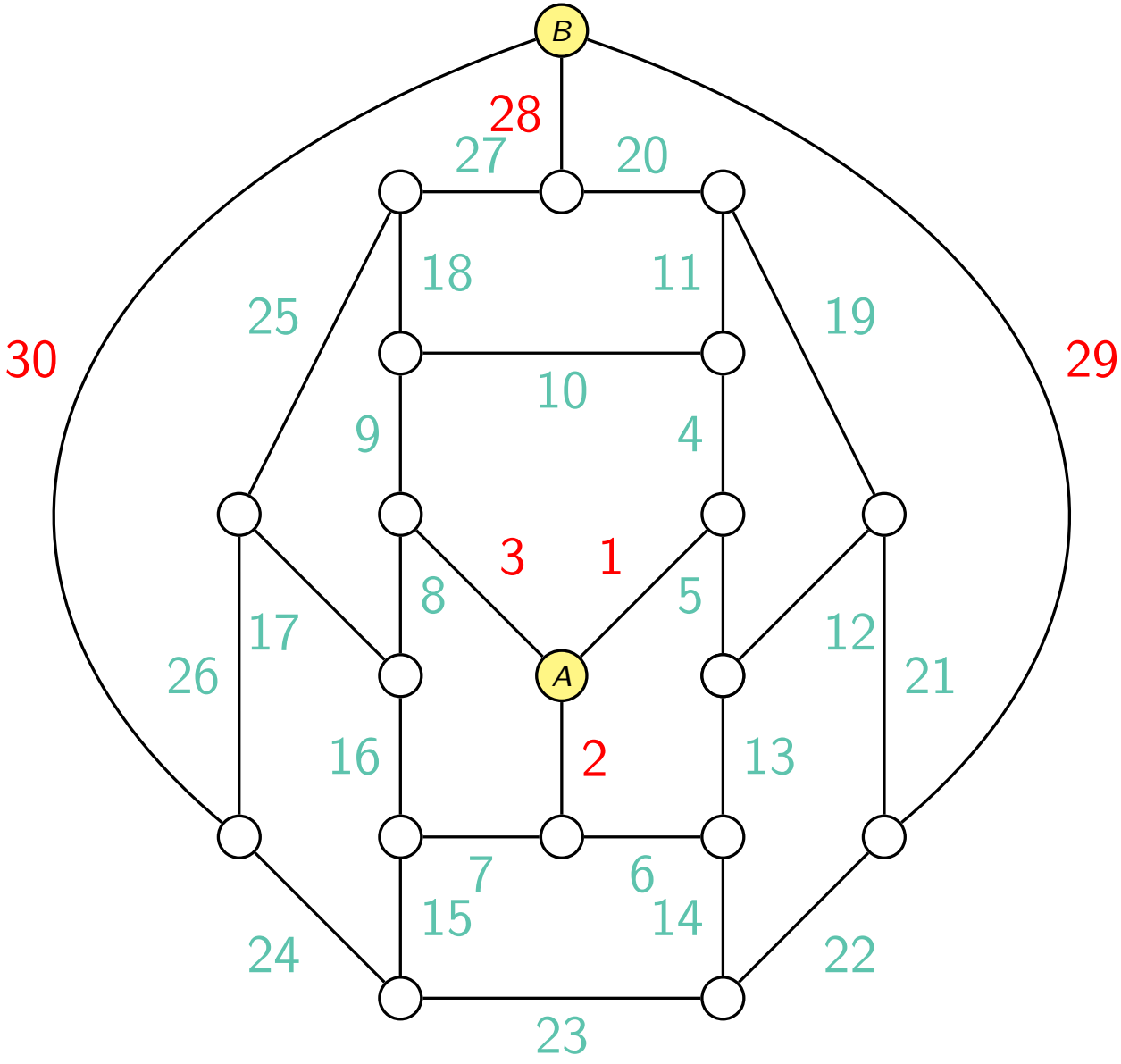
Can be developed as

$$\mathbb{P}\left[A_1 + \cdots + A_c > t \mid \pi,\, C = c\right] = \sum_{j=1}^{c} e^{-\Lambda_j t} \prod_{k=1,\, k\neq j}^{c} \frac{\Lambda_k}{\Lambda_k - \Lambda_j}.$$

This formula can quickly becomes unstable when some $\Lambda_k - \Lambda_j$ are small.

But Higham (2009) recently proposed a stable method for matrix exponential. Significantly slower but reliable.

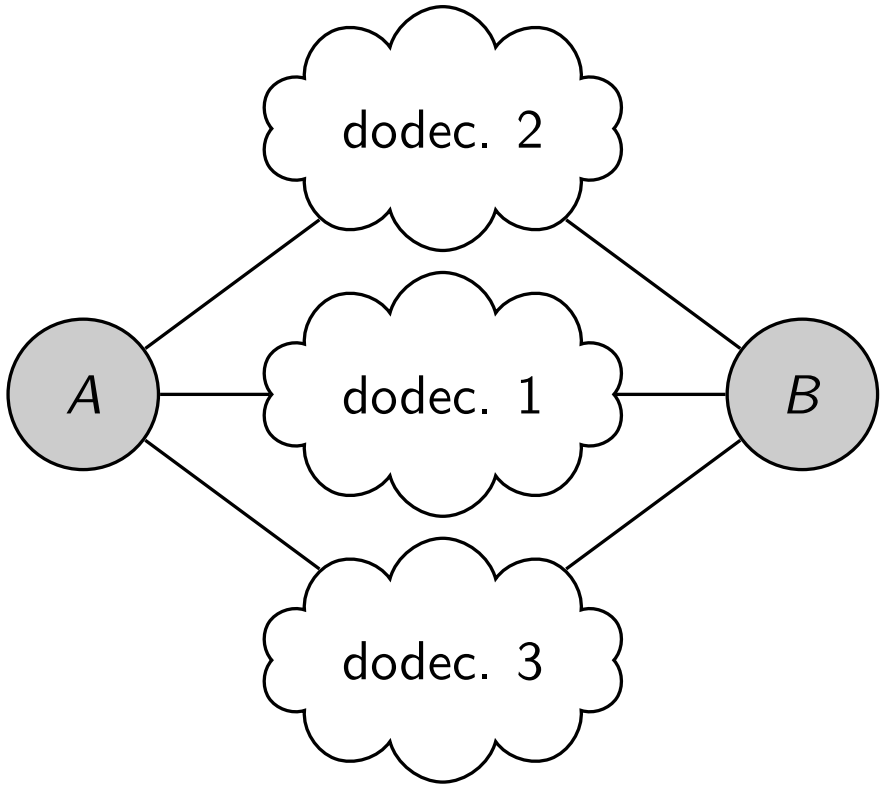
A dodecahedron network



Turnip method for dodecahedron graph: $n = 10^6$, $\mathcal{V}_0 = \{1, 20\}$

$q_i = \epsilon$	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}	10^{-6}
$\bar{W}_n$	2.881e-3	2.065e-6	2.006e-9	1.992e-12	1.999e-15	2.005e-18
RE$[\bar{W}_n]$	0.00302	0.00421	0.00433	0.00436	0.00435	0.00434
T (sec)	15.6	15.5	15.5	15.5	15.5	15.5

Three dodecahedron graphs in parallel.



Turnip for three dodecahedrons in parallel: $n = 10^8$, $\mathcal{V}_0 = \{1, 20\}$

$q_i = \epsilon$	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}	10^{-6}
$\bar{W}_n$	2.39e-8	8.80e-18	8.20e-27	8.34e-36	8.07e-45	7.92e-54
RE$[\bar{W}_n]$	0.0074	0.0194	0.0211	0.0210	0.0212	0.0215
T (sec)	6236	6227	6229	6546	6408	6289

Beats ZVA when ϵ is not too small.

Dependent Links: A Marshall-Olkin Copula Model

Goal: Define a model where the X_i's may have positive dependence.

Suppose all links are initially operational. For each $\mathbf{s} \subseteq \{1, \ldots, m\}$, a shock that takes down all links in $\mathbf{s}$ occurs at an exponential time with rate $\lambda_{\mathbf{s}}$. Let $\mathcal{L} = \{\mathbf{s} : \lambda_{\mathbf{s}} > 0\} = \{\mathbf{s}(1), \ldots, \mathbf{s}(\kappa)\}$, where $\kappa = |\mathcal{L}|$.

Denote $\lambda_j = \lambda_{\mathbf{s}(j)}$, let Y_j be the shock time for subset $\mathbf{s}(j)$, and $\mathbf{Y} = (Y_1, \ldots, Y_\kappa)$ (the latent state of the system).

Component i fails at time

$$W_i = \min\{Y_j : i \in \mathbf{s}(j) \text{ and } 1 \leq j \leq \kappa\},$$

so $X_i = \mathbb{I}[W_i > 1]$ the indicator that component i is operational at time 1.
We have $\mathbb{P}[X_i = 1] = \mathbb{P}[\min\{Y_j : i \in \mathbf{s}(j)\} > 1] = \exp\left[-\sum_{\{j:i\in\mathbf{s}(j)\}} \lambda_j\right]$.

Dependent Links: A Marshall-Olkin Copula Model

Goal: Define a model where the X_i's may have positive dependence.

Suppose all links are initially operational. For each $\mathbf{s} \subseteq \{1, \ldots, m\}$, a shock that takes down all links in $\mathbf{s}$ occurs at an exponential time with rate $\lambda_{\mathbf{s}}$. Let $\mathcal{L} = \{\mathbf{s} : \lambda_{\mathbf{s}} > 0\} = \{\mathbf{s}(1), \ldots, \mathbf{s}(\kappa)\}$, where $\kappa = |\mathcal{L}|$.

Denote $\lambda_j = \lambda_{\mathbf{s}(j)}$, let Y_j be the shock time for subset $\mathbf{s}(j)$, and $\mathbf{Y} = (Y_1, \ldots, Y_\kappa)$ (the latent state of the system).

Component i fails at time

$$W_i = \min\{Y_j : i \in \mathbf{s}(j) \text{ and } 1 \leq j \leq \kappa\},$$

so $X_i = \mathbb{I}[W_i > 1]$ the indicator that component i is operational at time 1.
We have $\mathbb{P}[X_i = 1] = \mathbb{P}[\min\{Y_j : i \in \mathbf{s}(j)\} > 1] = \exp\left[-\sum_{\{j:i\in\mathbf{s}(j)\}} \lambda_j\right]$.

Can have cascading shocks, to model cascading failure.

Problem: previous PMC and turnip methods do not apply directly.

PMC method, now a destruction process

Generate the shock times Y_j, sort them by increasing order to get $Y_{\pi(1)} < \cdots < Y_{\pi(\kappa)}$, and retain only the permutation π.

PMC estimator: $\mathbb{P}[\text{graph is failed at time 1} \mid \pi]$.

PMC method, now a destruction process

Generate the shock times Y_j, sort them by increasing order to get $Y_{\pi(1)} < \cdots < Y_{\pi(\kappa)}$, and retain only the permutation π.

PMC estimator: $\mathbb{P}[\text{graph is failed at time 1} \mid \pi]$.

To compute it, add the shocks $\pi(1)$, $\pi(2)$, ..., and remove corresponding links $i \in \mathbf{s}(j)$, until the system fails, say at shock C^* (the critical number).

Data structure: forest of spanning trees.
When removing a link: breath-first search for alternative path (costly).

The time $A_j = Y_{\pi(j)} - Y_{\pi(j-1)}$ between two successive shocks is exponential with rate Λ_j equal to the sum of rates of all forthcoming shocks. That is, $\Lambda_1 = \lambda_1 + \cdots + \lambda_\kappa$ and $\Lambda_{j+1} = \Lambda_j - \lambda_{\pi(j)}$ for $j \geq 1$.
PMC estimator of u: $\mathbb{P}\left[A_1 + \cdots + A_c \leq 1 \mid \pi, C^* = c\right]$.

Adapting the turnip method

Generate the shocks in increasing order of occurrence and at each step j, remove from consideration the future shocks that can no longer contribute to system failure.

For instance, when removing a link, if there are nodes that become disconnected from $\mathcal{V}_0$, those nodes can be removed for further consideration. And future shocks k that only affect removed links can be discarded, and their rate λ_k subtracted from Λ_j.

Adapting the turnip method

Generate the shocks in increasing order of occurrence and at each step j, remove from consideration the future shocks that can no longer contribute to system failure.

For instance, when removing a link, if there are nodes that become disconnected from $\mathcal{V}_0$, those nodes can be removed for further consideration. And future shocks k that only affect removed links can be discarded, and their rate λ_k subtracted from Λ_j.

Drawback remains: Removing links from the graph is more time consuming than adding links.

Scanning shocks in reverse order (construction process)

Once we have π, we can assume that all the shocks have occurred, go backward in time and remove them one by one, until the network is operational. This gives another way of computing C^*.

For each link i, initialize a counter $f(i)$ to the total number c_i of shocks that can can affect link i. Decrease the counter by 1 each time we remove a shock that affects link i. Link i is repaired when $f(i)$ reaches 0.

The network gets connected when removing (say) the C-th shock. Let $C^* = \kappa + 1 - C$ and compute the PMC estimator exactly as before.

PMC and turnip with antishocks (construction)

Here, to generate the permutation π, we generate the shock times directly in reverse order, i.e., we generate antishocks in increasing order.

Antishock j occurs at exponential time R_j with rate μ_j. We must have $1 - e^{-\mu_j} = \mathbb{P}[R_j \le 1] = \mathbb{P}[Y_j > 1] = e^{-\lambda_j}$, therefore $\mu_j = -\ln(1 - e^{-\lambda_j})$.

If the system gets connected at the C-th antishock, the PMC estimator is $\mathbb{P}\left[A_1 + \cdots + A_c > 1 \mid \pi, C = c\right]$ where each A_j is exponential with rate $\Lambda_j = \mu_{\pi(j)} + \cdots + \mu_{\pi(\kappa)}$.

PMC and turnip with antishocks (construction)

Here, to generate the permutation π, we generate the shock times directly in reverse order, i.e., we generate antishocks in increasing order.

Antishock j occurs at exponential time R_j with rate μ_j. We must have $1 - e^{-\mu_j} = \mathbb{P}[R_j \le 1] = \mathbb{P}[Y_j > 1] = e^{-\lambda_j}$, therefore $\mu_j = -\ln(1 - e^{-\lambda_j})$.

If the system gets connected at the C-th antishock, the PMC estimator is $\mathbb{P}\left[A_1 + \cdots + A_c > 1 \mid \pi, C = c\right]$ where each A_j is exponential with rate $\Lambda_j = \mu_{\pi(j)} + \cdots + \mu_{\pi(\kappa)}$.

Turnip version: eliminate antishocks that become useless. Adds overhead.

Dodecahedron, shocks on links only, $q_j = q = 10^{-3}$, $\mathcal{V}_0 = \{1, 20\}$, $n = 10^6$.

Algo:	PMC shocks	PMC reverse	turnip shocks	turnip reverse	turnip anti	GS shocks
$\bar{W}_n$	2.009e-9	2.009e-9	1.9603e-9	2.067e-9	1.800e-9	2.020e-9
$S_n^2/(\bar{W}_n)^2$	1932.68	1932.68	1980.71	1881.09	2153.01	58.4304
RE[$\bar{W}_n$]	0.04396	0.04396	0.04450	0.04337	0.0464	0.00764
C	13.28	13.28	13.279	13.2789	17.72	—
T (sec)	29.2	20.2	26.3	20.5	8.4	176.5
WNRV[$\bar{W}_n$]	0.0565	0.0390	0.0520	0.0385	0.0181	0.0103

WRNV = Work-normalized relative variance.

For turnip-anti, we use (and give) C^* instead of C.

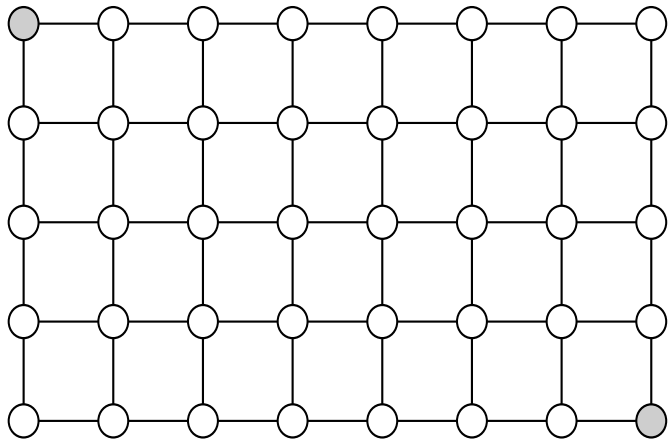
Complete graph with 50 nodes.

One shock per node, kills all links connected to that node.

Rate $\lambda_j = 10^{-3}$, $\mathcal{V}_0 = \{1, 50\}$, $n = 10^5$.

Algo:	PMC shocks	PMC reverse	turnip shocks	turnip reverse	turnip anti	GS shocks
$\bar{W}_n$	0.001968	0.001968	0.002020	0.001988	0.001991	0.001939
$S_n^2/(\bar{W}_n)^2$	23.2138	23.2138	22.5845	22.9627	22.9417	17.5215
RE$[\bar{W}_n]$	0.015236	0.015236	0.015028	0.015153	0.015147	0.013237
C	16.99	16.99	16.94	16.98	34.00	—
T (sec)	199.3	7.1	189.1	9.9	7.5	343.0
WNRV$[\bar{W}_n]$	0.0463	0.00166	0.0427	0.00228	0.00173	0.0601

A lattice graph



20×20 square lattice graph

400 nodes and 760 links. One shock per node at rate $\lambda = 10^{-5}$ and one shock per link at rate $10\lambda = 10^{-4}$. $\mathcal{V}_0 = \{1, 400\}$, $n = 10^4$.

Algo:	PMC shocks	PMC reverse	turnip shocks	turnip reverse	turnip anti	GS shocks
$\bar{W}_n$	1.5999e-5	1.5999e-5	1.5999e-5	1.5999e-5	1.7198e-5	2.0056e-5
$S_n^2/(\bar{W}_n)^2$	248.055	248.055	248.055	248.055	230.699	24.7837
RE$[\bar{W}_n]$	0.157498	0.157498	0.157498	0.157498	0.151888	0.049783
T (sec)	48.1	12.4	48.8	12.6	8.1	93.2
WNRV$[\bar{W}_n]$	1.19	0.306	1.21	0.312	0.188	0.231

In this example, whenever we decrease λ or increase the number of nodes, PMC and turnip fail totally (they give wrong estimates). Similar results if we put shocks only on links. Here, C is near 240 on average.

The important permutations π become rare and often not seen.
We will use an adaptive learning algorithm (GS) to find them.

A generalized splitting (GS) algorithm

Uses latent variables $\mathbf{Y}$. Let

$$\tilde{S}(\mathbf{Y}) = \inf\{\gamma \geq 0 : \Psi(\mathbf{X}(\gamma)) = 0\},$$

the time at which the network fails, and $S(\mathbf{Y}) = 1/\tilde{S}(\mathbf{Y})$.

Choose real numbers $0 = \gamma_0 < \gamma_1 < \cdots < \gamma_\tau = 1$ for which

$$\rho_t \stackrel{\text{def}}{=} \mathbb{P}[S(\mathbf{Y}) > \gamma_t \mid S(\mathbf{Y}) > \gamma_{t-1}] \approx 1/2$$

for $t = 1, \ldots, \tau$.

For each level γ_t, construct (via MCMC) a Markov chain $\{\mathbf{Y}_{t,j}, j \geq 0\}$ with transition density κ_t and whose stationary density is the density of $\mathbf{Y}$ conditional on $S(\mathbf{Y}) > \gamma_t$:

$$f_t(\mathbf{y}) \stackrel{\text{def}}{=} f(\mathbf{y}) \frac{\mathbb{I}[S(\mathbf{y}) > \gamma_t]}{\mathbb{P}[S(\mathbf{Y}) > \gamma_t]}.$$

GS algorithm

Generate $\mathbf{Y}$ from density f
if $S(\mathbf{Y}) > \gamma_1$ **then** $\mathcal{X}_1 \leftarrow \{\mathbf{Y}\}$ **else return** $U \leftarrow 0$
for $t = 2$ **to** τ **do**
    $\mathcal{X}_t \leftarrow \emptyset$ // set of states that have reached level γ_t
    for all $\mathbf{Y}_0 \in \mathcal{X}_{t-1}$ **do**
        for $\ell = 1$ **to** 2 **do**
            sample $\mathbf{Y}_\ell$ from density $\kappa_{t-1}(\cdot \mid \mathbf{Y}_{\ell-1})$
            if $S(\mathbf{Y}_\ell) > \gamma_t$ **then** add $\mathbf{Y}_\ell$ to $\mathcal{X}_t$
return $U \leftarrow |\mathcal{X}_\tau| / 2^{\tau-1}$ as an unbiased estimator of u.

Defining κ_{t-1} via Gibbs sampling:

Require: $\mathbf{Y}$ for which $S(\mathbf{Y}) > \gamma_{t-1}$ and a permutation π of $\{1, \ldots, \kappa\}$

  for $k = 1$ **to** κ **do**

    $j \leftarrow \pi(k)$

    if $S(Y_1, \ldots, Y_{j-1}, 0, Y_{j+1}, \ldots, Y_\kappa) < \gamma_{t-1}$ **then**

      // removing shock j would connect $\mathcal{V}_0$

      resample Y_j from its density truncated to $(0, 1/\gamma_{t-1})$

    else

      resample Y_j from its original density

  return $\mathbf{Y}$ as the resampled vector.

Defining κ_{t-1} via Gibbs sampling:

Require: $\mathbf{Y}$ for which $S(\mathbf{Y}) > \gamma_{t-1}$ and a permutation π of $\{1, \ldots, \kappa\}$
  for $k = 1$ **to** κ **do**
    $j \leftarrow \pi(k)$
    if $S(Y_1, \ldots, Y_{j-1}, 0, Y_{j+1}, \ldots, Y_\kappa) < \gamma_{t-1}$ **then**
      // removing shock j would connect $\mathcal{V}_0$
      resample Y_j from its density truncated to $(0, 1/\gamma_{t-1})$
    else
      resample Y_j from its original density
  return $\mathbf{Y}$ as the resampled vector.

How to estimate appropriate levels γ_t? Adaptive (pilot) phase.

Data structure: forest of spanning trees.

GS for the dodecahedron, shocks on links only: $n = 10^6$, $\mathcal{V}_0 = \{1, 20\}$

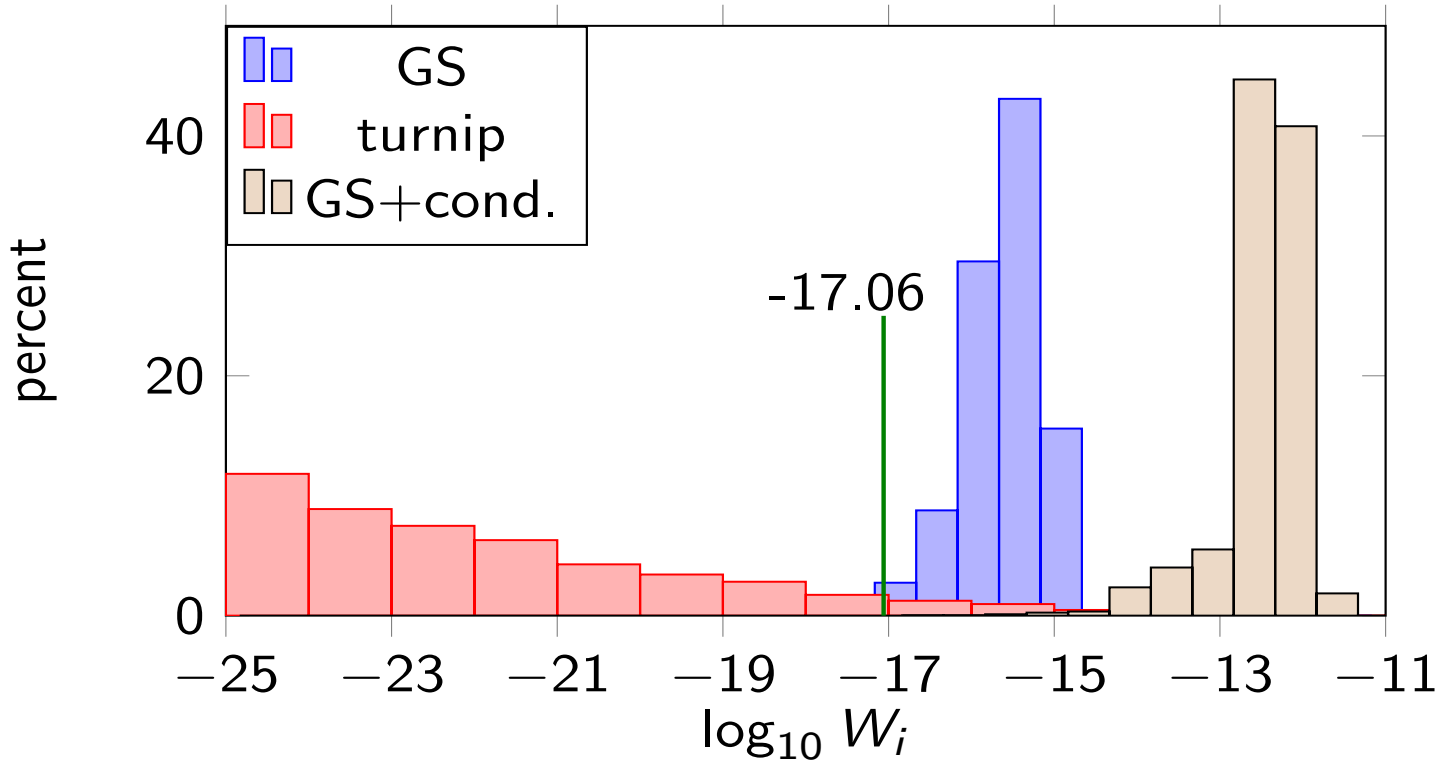
$q_j = \epsilon$	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}	10^{-6}
τ	9	19	29	39	49	59
$\bar{W}_n$	0.002877	2.054e-6	2.022e-9	2.01e-12	1.987e-15	1.969e-18
RE$[\bar{W}_n]$	0.00403	0.0062	0.00769	0.0089	0.00992	0.0112
T (sec)	93	167	224	278	334	376

GS
for the three dodecahedrons in parallel, shocks on links only: $n = 10^6$, $\mathcal{V}_0 = \{1, 20\}$

$q_j = \epsilon$	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}	10^{-6}
τ	26	57	87	117	147	176
$\bar{W}_n$	2.377e-8	8.874e-18	8.182e-27	8.088e-36	8.237e-45	7.931e-54
RE$[\bar{W}_n]$	0.00712	0.0109	0.0137	0.0158	0.0185	0.0208
T (sec)	1202	2015	2362	2820	3041	3287

Dodecahedron: distribution of states at last level

Histograms of $\log_{10}(U)$ for GS (middle), turnip (left), and for the conditional prob. of failure for the permutations π obtained by GS (right), for three dodecahedrons in parallel, with $q = 10^{-2}$.



Square lattice graph, shocks on links only

GS for a 50×50 lattice graph, with 2500 nodes, 4900 links, $\mathbb{P}[X_i = 0] = \epsilon$, $n = 10^4$.

$q_j = \epsilon$	10^{-2}	10^{-3}	10^{-4}	10^{-5}	10^{-6}
τ	13	19	26	33	39
$\bar{W}_n$	2.148e-4	2.085e-6	2.179e-8	2.156e-10	1.932e-12
RE$[\bar{W}_n]$	0.0466	0.0604	0.0678	0.0785	0.0909
T (sec)	19818	19283	18413	17967	17851

PMC and turnip are useless in this case.

20×20 square lattice graph

400 nodes and 760 links. One shock per node at rate $\lambda = 10^{-5}$ and one shock per link at rate $10\lambda = 10^{-4}$. $\mathcal{V}_0 = \{1, 400\}$, $n = 10^4$.

Algo:	PMC shocks	PMC reverse	turnip shocks	turnip reverse	turnip anti	GS shocks
$\bar{W}_n$	1.5999e-5	1.5999e-5	1.5999e-5	1.5999e-5	1.7198e-5	2.0056e-5
$S_n^2/(\bar{W}_n)^2$	248.055	248.055	248.055	248.055	230.699	24.7837
RE$[\bar{W}_n]$	0.157498	0.157498	0.157498	0.157498	0.151888	0.049783
T (sec)	48.1	12.4	48.8	12.6	8.1	93.2
WNRV$[\bar{W}_n]$	1.19	0.306	1.21	0.312	0.188	0.231

20×20 lattice graph, 400 nodes and 760 links.
One shock per node at rate λ and one shock per link at rate 10λ.
$\mathcal{V}_0 = \{1, 400\}$, GS with shocks, $n = 10^4$.

λ	$\bar{W}_n$	RE$[\bar{W}_n]$	T (sec)
10^{-2}	4.66e-2	0.0283	102
10^{-3}	2.16e-3	0.0480	133
10^{-4}	2.00e-4	0.0624	122
10^{-5}	1.95e-5	0.0629	153
10^{-6}	2.17e-6	0.0653	168
10^{-7}	2.14e-7	0.0634	184
10^{-8}	2.05e-8	0.1203	105
10^{-9}	1.97e-9	0.1093	150
10^{-10}	1.94e-10	0.0696	266
10^{-11}	1.97e-11	0.0819	187
10^{-12}	2.16e-12	0.0629	359
10^{-18}	1.93e-18	0.0712	811

PMC and turnip do not work here when λ is too small.

Extensions

PMC, turnip, and GS could be adapted to rare-event simulation in even more general shock-based reliability models, e.g., where shocks only alter the state of the system, may change the future shock rates, etc. Several applications in sight.