Some Rare-Event Simulation Methods for Static Network Reliability Estimation

Pierre L'Ecuyer
Université de Montréal, Canada

based on joint work with

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

Summer School in Monte Carlo Methods for Rare Events
Brown University, June 2016

Aim

Introduce and illustrate some rare-event simulation ideas that are less standard but have potential, via a simple application.

- Conditional Monte Carlo with auxiliary variables.
- Splitting when splitting does not seem to apply.
- Strategies for approximate zero-variance importance sampling.

A static system reliability problem

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

A static system reliability problem

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

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

$$u = \sum_{\mathbf{x} \in \mathcal{D} = \{\mathbf{X}:\Phi(\mathbf{X})=0\}} p(\mathbf{x}).$$

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

The X_i's may be dependent.

Monte Carlo (MC): 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
$\bar{W}_n = (W_1 + \cdots + W_n)/n \sim \text{Binomial}(n, u)/n \approx \text{Poisson}(nu)/n$.
Can also estimate $\text{Var}[\bar{W}_n]$ and compute a confidence interval on u.

Monte Carlo (MC): 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
$\bar{W}_n = (W_1 + \cdots + W_n)/n \sim \text{Binomial}(n, u)/n \approx \text{Poisson}(nu)/n$.
Can also estimate $\text{Var}[\bar{W}_n]$ and compute a confidence interval on u.

When u is very small (failure is a rare event), direct MC fails.
Ex: if $u = 10^{-10}$, system fails once per 10 billion runs on average.

Monte Carlo (MC): 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
$\bar{W}_n = (W_1 + \cdots + W_n)/n \sim \text{Binomial}(n, u)/n \approx \text{Poisson}(nu)/n$.
Can also estimate $\text{Var}[\bar{W}_n]$ and compute a confidence interval on u.

When u is very small (failure is a rare event), direct MC fails.
Ex: if $u = 10^{-10}$, system fails once per 10 billion runs on average.

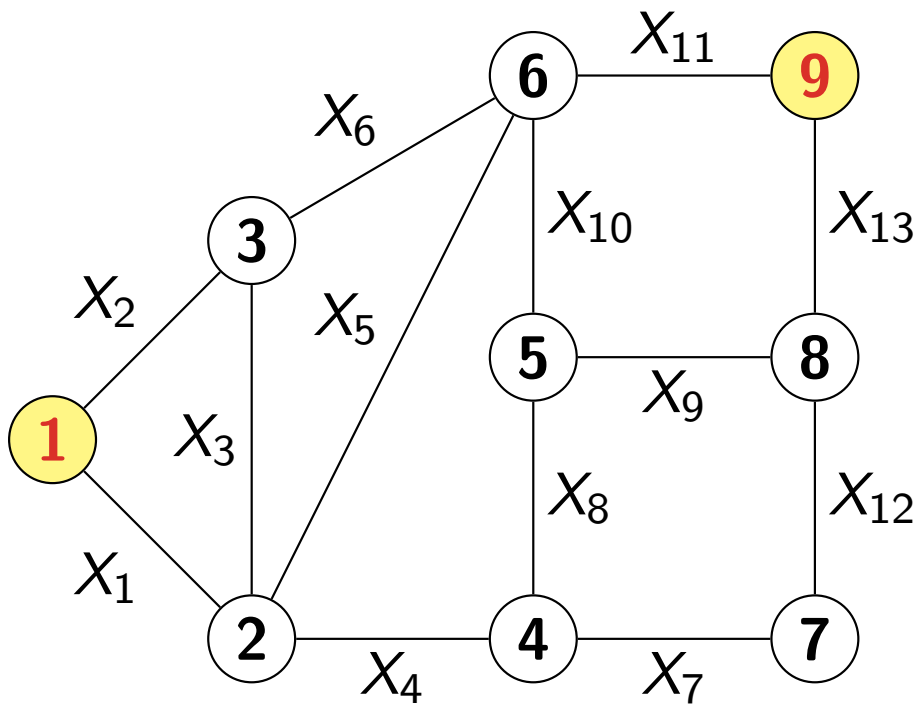
Relative error

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

For example, if $u \approx 10^{-10}$, we need $n \approx 10^{12}$ to have $\text{RE}[\bar{W}_n] \leq 10\%$.

We would like bounded RE (or almost) when $u \to 0$.

Although our methods apply more generally, we focus here on graph reliability. Link i "works" iff $X_i = 1$.
The system is operational iff all the nodes in a given set $\mathcal{V}_0$ are connected.



Given $\mathbf{X}$, $\Phi(\mathbf{X})$ is easy to evaluate by graph algorithms.
Challenge: How to sample $\mathbf{X}$ effectively.

Conditional Monte Carlo

Idea: replace an estimator X by $\mathbb{E}[X \mid \mathcal{G}]$ for a σ-field $\mathcal{G}$ that contains partial information on X. The CMC estimator is $X_{\mathrm{e}} \stackrel{\mathrm{def}}{=} \mathbb{E}[X \mid \mathcal{G}]$.

We have $\mathbb{E}[X_{\mathrm{e}}] = \mathbb{E}[\mathbb{E}[X \mid \mathcal{G}]] = \mathbb{E}[X]$ and

$$\mathrm{Var}[X] = \mathbb{E}[\underbrace{\mathrm{Var}[X \mid \mathcal{G}]}_{\text{Residual variance when } \mathcal{G} \text{ is known (eliminated by CMC)}}] + \underbrace{\mathrm{Var}[\mathbb{E}[X \mid \mathcal{G}]]}_{\text{Var due to the variation of } \mathcal{G}} = \mathbb{E}[\mathrm{Var}[X \mid \mathcal{G}]] + \mathrm{Var}[X_{\mathrm{e}}].$$

Therefore (Rao-Blackwell theorem):

$$\mathrm{Var}[X_{\mathrm{e}}] \;=\; \mathrm{Var}[X] - \mathbb{E}[\mathrm{Var}[X \mid \mathcal{G}]] \;\leq\; \mathrm{Var}[X].$$

To maximize $\mathbb{E}[\mathrm{Var}[X \mid \mathcal{G}]]$, $\mathcal{G}$ should contain as little information as possible, but then computing X_{e} may become too hard. The choice of $\mathcal{G}$ is a matter of compromise.

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.

Time to repair, conditional on π?
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.
Permutation Monte Carlo (PMC) estimator of u: conditional probability that the total time for these repairs (hypoexponential sum) is larger than 1:

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

Theorem [Gertsback and Shpungin 2010]. Gives bounded RE when the $q_i \to 0$.

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.

Time to repair, conditional on π?
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.
Permutation Monte Carlo (PMC) estimator of u: conditional probability that the total time for these repairs (hypoexponential sum) is larger than 1:

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

Theorem [Gertsback and Shpungin 2010]. Gives bounded RE when the $q_i \to 0$.

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

Hypoexponential cdf: We have

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

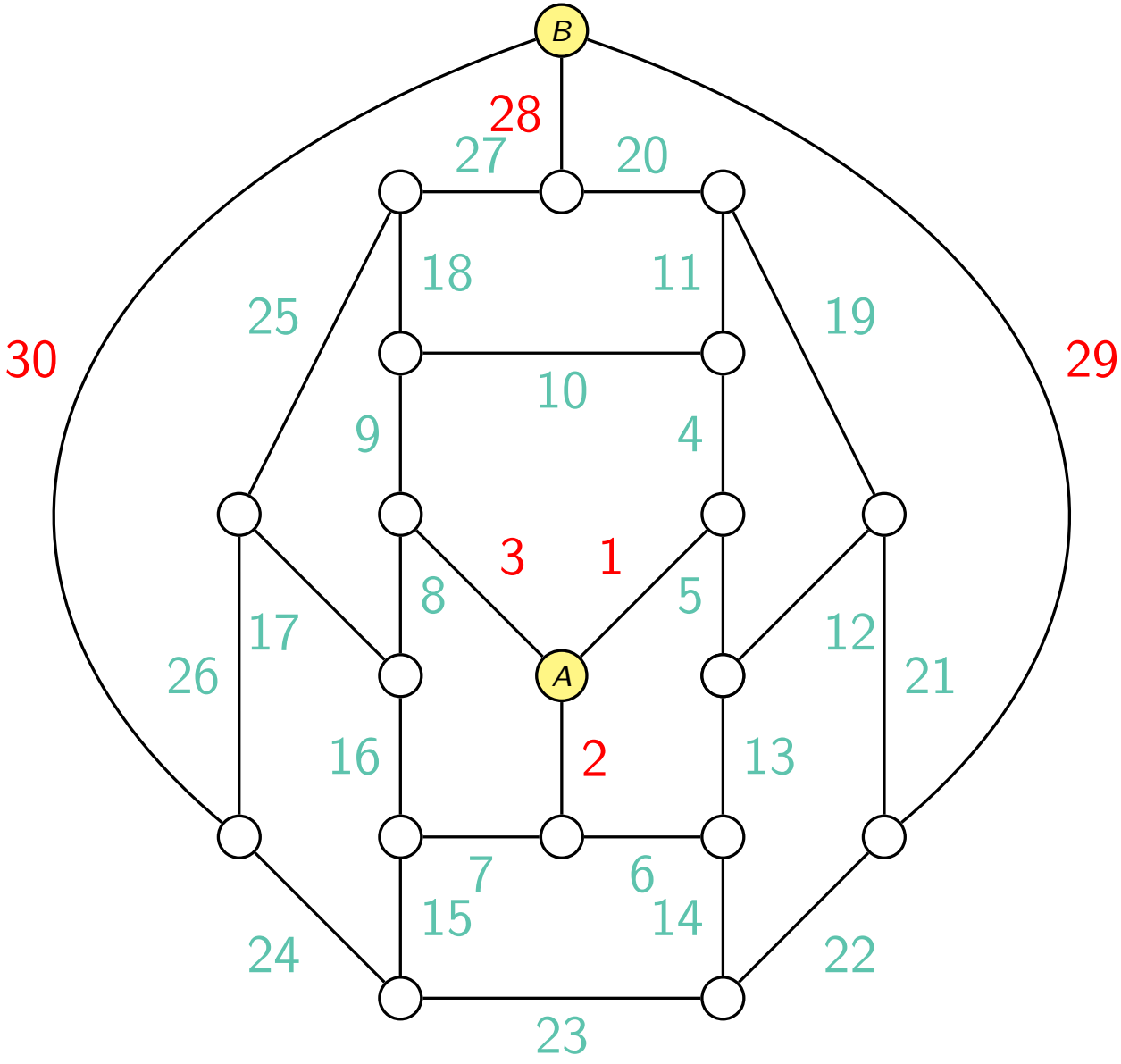
This formula becomes unstable when c is large and/or the Λ_j are small. The product terms are very large and have alternate signs $(-1)^{j-1}$.

Higham (2009) proposes a stable method for matrix exponential. More reliable, but significantly slower.

For the case where the above prob is close to 1, we also have

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

A dodecahedron network



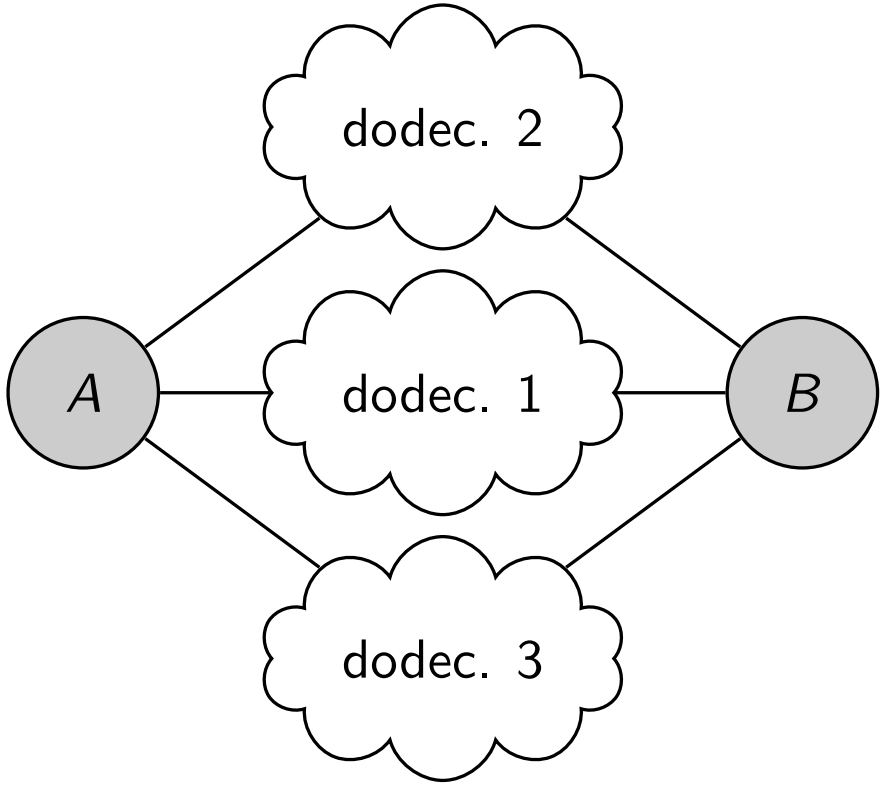
Turnip method for dodecahedron graph: $n = 10^6$, $\mathcal{V}_0 = \{1, 20\}$

$q_i = \epsilon$	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

We see that $u \approx 2 \times 10^{-3\epsilon}$ and RE is bounded (proved).

Three dodecahedron graphs in parallel.

60 nodes and 90 links.



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

We have $u \approx 2 \times 10^{-9\epsilon}$ and RE is bounded (proved).
Total CPU time is about 2 hours, regardless of ϵ.

However, for very large graphs (thousands of links), the turnip method fails, because the important permutations π, for which the conditional probability contributes significantly, are rare, and hitting them becomes a rare event.

Bounded RE does not hold for an asymptotic regime where the size of the graph increases. Splitting will come to the rescue (later on).

Dependent Links: A Marshall-Olkin Copula Model

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

Dependent Links: A Marshall-Olkin Copula Model

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

We use an auxiliary dynamic model to specify the 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 $\mathcal{L} = \{\mathbf{s} : \lambda_{\mathbf{s}} > 0\} = \{\mathbf{s}(1), \ldots, \mathbf{s}(\kappa)\}$.
This can represent group failures and cascading failures (quite natural).

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).

X_i is the the indicator that component i is operational at time 1:

$$X_i = \mathbb{I}[Y_j > 1 \text{ for all shocks } j \text{ such that } i \in \mathbf{s}(j)\}.$$

Dependent Links: A Marshall-Olkin Copula Model

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

We use an auxiliary dynamic model to specify the 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 $\mathcal{L} = \{\mathbf{s} : \lambda_{\mathbf{s}} > 0\} = \{\mathbf{s}(1), \ldots, \mathbf{s}(\kappa)\}$.
This can represent group failures and cascading failures (quite natural).

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).

X_i is the the indicator that component i is operational at time 1:

$$X_i = \mathbb{I}[Y_j > 1 \text{ for all shocks } j \text{ such that } i \in \mathbf{s}(j)\}.$$

The previous PMC and turnip methods do not apply here, because the "repairs" or failures of links are not independent!

PMC method, now a destruction process

Generate the shock times Y_j (instead of link failure or repair times), sort them 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 (instead of link failure or repair times), sort them 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, at critical shock number C_{s}.

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

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:

$$U = \mathbb{P}\left[A_1 + \cdots + A_c \leq 1 \mid \pi, C_{\mathrm{s}} = c\right] = \sum_{j=1}^{c} (1 - e^{-\Lambda_j}) \prod_{k=1,\, k\neq j}^{c} \frac{\Lambda_k}{\Lambda_k - \Lambda_j}.$$

Generating the permutation π directly

At step k, the kth shock is selected with probability λ_j/Λ_k for shock j, where Λ_k is the sum of rates for the shocks that remain. This avoids the sort, and we stop when we reach C_{s}.

However, the probabilities λ_j/Λ_k change at each step, so they must be updated to generate the next shock. Could bring significant overhead: $\mathcal{O}(\kappa)$ time at each step; $\mathcal{O}(C_{\mathrm{s}}\kappa)$ time overall. So it is slower in some situations.

A special case: If the λ_j are all equal, the next shock is always selected uniformly. This amounts to generating a random permutation, which is easy to do efficiently.
We also have a formula to compute the hypoexponential cdf must faster in this case.

Scanning the shocks in reverse order

Instead of adding shocks until the system fails, we can generate all the shocks to know π, then assume that all shocks have already occurred, and remove them one by one until $\mathcal{V}_0$ is connected. Reconstructing the network like this is sometimes much faster.

If c_i shocks can affect link i, start a counter f_i at c_i, and decrease it each time a shock that affects i is removed. Link i is repaired when $f_i = 0$.

C_{S} is the number of shocks that remain when the system becomes operational, plus 1.

This gives a faster way to compute C_{S} when it is large (close to κ). The estimator U remains the same.

PMC with anti-shocks

Here we change the estimator. Assume all the shocks have occurred and generate independent anti-shocks that remove the shocks, one by one. Idea: repair the shocks rather than the links.

Anti-shock j occurs at exponential time R_j, with rate $\mu_j = -\ln(1 - e^{-\lambda_j})$. This gives $\mathbb{P}[R_j \le 1] = \mathbb{P}[Y_j > 1] = \mathbb{P}[\text{shock } j \text{ has occurred}]$.

Sorting the times R_j gives a permutation π' ($\equiv$ reverse of π).
$C_a = \kappa + 1 - C_s =$ anti-shock number when system becomes operational.
Times between successive anti-shocks: $A'_k = R_{\pi'(k)} - R_{\pi'(k-1)}$, exponential with rate $\Lambda_k = \mu_{\pi(k)} + \cdots + \mu_{\pi(\kappa)}$. Estimator of u:

$$U' = \mathbb{P}[A'_1 + \cdots + A'_{C_a} > 1 \mid \pi'].$$

When u is very small, we can often compute U' accurately and not U.

Adapting the turnip method

When generating the shocks [or anti-shocks] in increasing order of occurrence, at each step j, discard the future shocks [or anti-shocks] that can no longer contribute to system failure [or repair].

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

When generating the shocks [or anti-shocks] in increasing order of occurrence, at each step j, discard the future shocks [or anti-shocks] that can no longer contribute to system failure [or repair].

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.

When an anti-shock occurs, if it repairs a link that connects two groups of nodes, all links that connect the same groups can be discarded, and anti-shocks that only affect discarded links can be discarded.

Overhead: Must maintain data structures to identify shocks [or anti-shocks] that can be discarded.

Bounded relative error for PMC and turnip

Under mild conditions, we can prove that the PMC and turnip estimators have bounded RE when the $\lambda_j \to 0$, for a fixed graph.

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 $S(\mathbf{Y}) = 1/\tilde{S}(\mathbf{Y})$.
We want samples of $\mathbf{Y}$ for which $S(\mathbf{Y}) > 1$.

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 $t = 1, \ldots, \tau$. The γ_t's are estimated by pilot runs.

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 $S(\mathbf{Y}) > \gamma_t$:

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

Defining κ_{t-1} via Gibbs sampling:

Require: $\mathbf{Y}$ for which $S(\mathbf{Y}) > \gamma_{t-1}$
  for $j = 1$ **to** κ **do**
    if $S(Y_1, \ldots, Y_{j-1}, \infty, Y_{j+1}, \ldots, Y_\kappa) < \gamma_{t-1}$ **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.

Data structure: forest of spanning trees.

GS algorithm with shocks

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.

Repeat this n times, independently, and take the average.
Can compute a confidence interval, etc.

GS algorithm with anti-shocks

Same idea, but evolution and resampling is based on $\mathbf{R}$ instead of $\mathbf{Y}$.

$$S(\mathbf{R}) = \inf\{\gamma \geq 0 : \Psi(\mathbf{X}(\gamma)) = 1\}.$$

Generate a vector $\mathbf{R}$ of anti-shock times from its unconditional density.
if $S(\mathbf{R}) > \gamma_1$ **then**
    $\mathcal{X}_1 \leftarrow \{\mathbf{R}\}$
else
    return $U \leftarrow 0$
for $t = 2$ **to** τ **do**
    $\mathcal{X}_t \leftarrow \emptyset$ // states that have reached level γ_t
    for all $\mathbf{R}_0 \in \mathcal{X}_{t-1}$ **do**
        for $\ell = 1$ **to** 2 **do**
            sample $\mathbf{R}_\ell$ from the density $\kappa_{t-1}(\cdot \mid \mathbf{R}_{\ell-1})$
            if $S(\mathbf{R}_\ell) > \gamma_t$ **then**
                add $\mathbf{R}_\ell$ to $\mathcal{X}_t$
return $U \leftarrow |\mathcal{X}_\tau|/2^{\tau-1}$, an unbiased estimate of u.

Gibbs sampling for anti-shocks density $\kappa_{t-1}(\cdot \mid \mathbf{R})$:

Require: $\mathbf{R} = (R_1, \ldots, R_\kappa)$ for which $S(\mathbf{R}) > \gamma_{t-1}$.
  for $j = 1$ **to** κ **do**
    if $S(R_1, \ldots, R_{j-1}, 0, R_{j+1}, \ldots, R_\kappa) \leq \gamma_{t-1}$ **then**
      resample R_j from its density truncated to (γ_{t-1}, ∞)
    else
      resample R_j from its original density
  return $\mathbf{R}$ as the resampled vector.

Example: dodecahedron graph

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
$\bar{W}_n$	2.877e-3	2.054e-6	2.022e-9	2.01e-12	1.987e-15	1.969e-18
RE$[\bar{W}_n]$	0.0040	0.0062	0.0077	0.0089	0.0099	0.0112
T (sec)	93	167	224	278	334	376

GS, three dodeca. in parallel, shocks on links: $n = 10^6$, $\mathcal{V}_0 = \{1, 20\}$

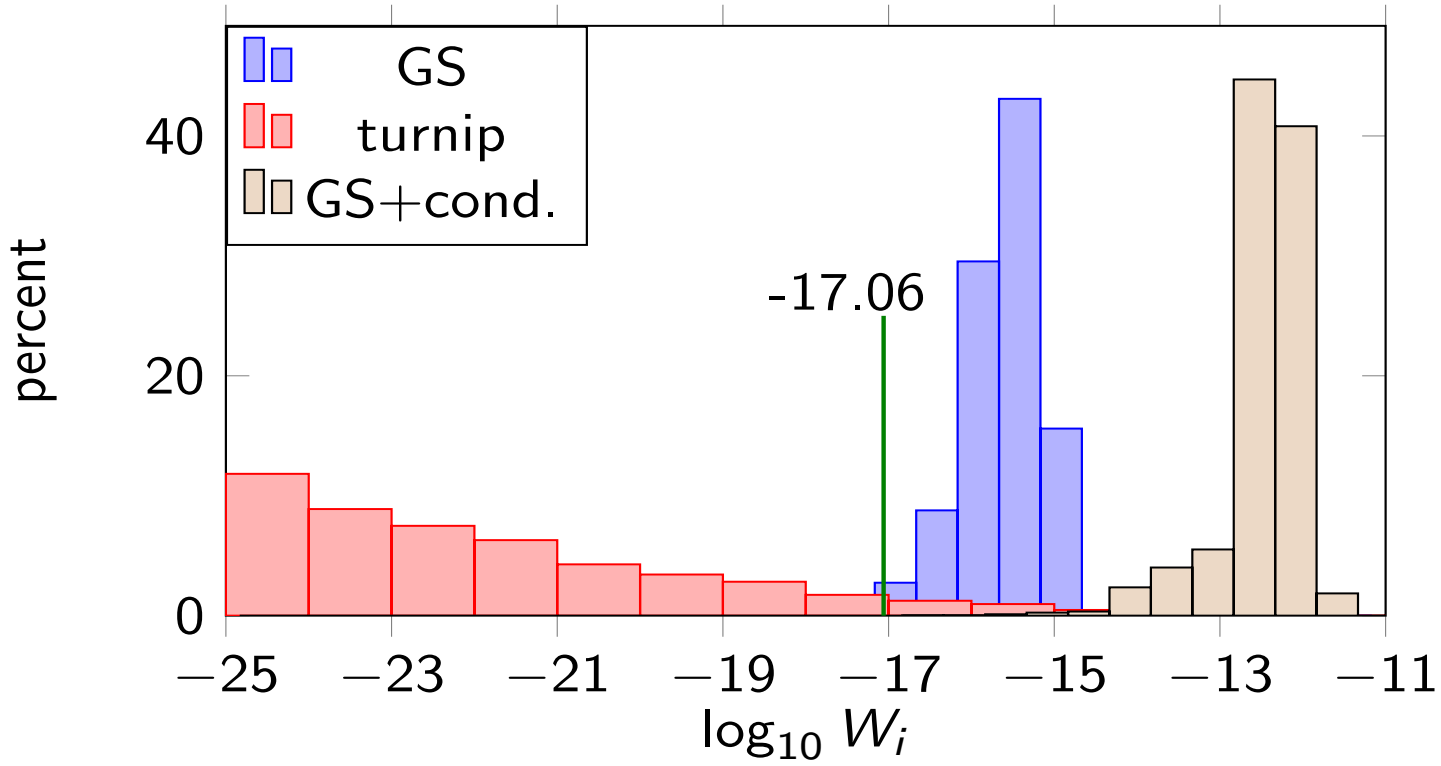
$q_j = \epsilon$	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}	10^{-6}
τ	26	57	87	117	147	176
$\bar{W}_n$	2.38e-8	8.87e-18	8.18e-27	8.09e-36	8.24e-45	7.93e-54
RE$[\bar{W}_n]$	0.0071	0.0109	0.0137	0.0158	0.0185	0.0208
T (sec)	1202	2015	2362	2820	3041	3287

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

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 $q = 10^{-2}$.



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

We see that $u \approx 2 \times 10^{-3\epsilon}$ and RE is bounded (proved).

Dodecahedron graph, shocks on nodes and on links

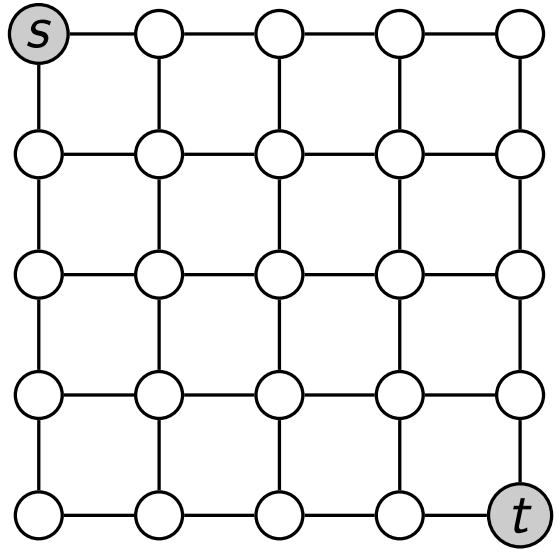
All shocks at rate λ except on $\mathcal{V}_0 = \{1, 20\}$; $n = 10^6$.

algorithm	$\bar{W}_n$	$S_n^2/\bar{W}_n^2$	RE$[\bar{W}_n]$	$\bar{C}$	T(sec)	WNRV
			$\lambda = 10^{-3}$			
PMC	1.62e-8	993	0.032	12.7	35	0.035
PMC-anti	1.60e-8	1004	0.032	36.3	17	0.018
turnip	1.63e-8	894	0.030	10.7	• 72	0.064
turnip-anti	1.58e-8	296	0.017	35.8	◦ 45	0.013
GS	1.59e-8	53	0.007		437	0.023
GS-anti	1.60e-8	56	0.007		425	0.024
			$\lambda = 10^{-7}$			
PMC	1.65e-20	1047	0.032	12.7	32	0.034
PMC-anti	1.66e-20	1044	0.032	36.3	18	0.019
turnip-anti	1.58e-20	311	0.018	35.8	◦ 44	0.014
GS	1.59e-20	143	0.012		982	0.140
GS-anti	1.58e-20	124	0.011		1106	0.137

Three dodecahedrons in parallel, $n = 10^6$

algorithm	$\bar{W}_n$	$S_n^2/\bar{W}_n^2$	RE$[\bar{W}_n]$	$\bar{C}$	T(sec)	WNRV
			$\lambda = 0.1$			
pmc	1.79e-5	3157	0.056	50	207	0.66
pmc-anti	1.72e-5	2410	0.049	95	52	0.13
turn	1.77e-5	2572	0.051	38	• 771	1.98
turn-anti	1.73e-5	1473	0.038	94	◦ 215	0.32
GS	1.79e-5	31	0.0056		1094	0.034
GS-anti	1.78e-5	30	0.0055		1141	0.034
			$\lambda = 0.001$			
turn-anti	1.20e-29	5.7e5	0.75	94	◦ 216	12
GS	4.13e-24	158	0.013		4366	0.70
GS-anti	4.06e-24	197	0.014		3552	0.70

Square lattice graphs



20×20 lattice: 400 nodes, 760 links, and 1158 different shocks.
40×40 lattice: 1600 nodes, 3120 links, and 4718 different shocks.

20×20 lattice graph, $n = 10^5$

algorithm	$\bar{W}_n$	$S_n^2/\bar{W}_n^2$	RE$[\bar{W}_n]$	$\bar{C}$	T(sec)	WNRV
			$\lambda = 10^{-5}$			
PMC	6.67e-10	9.9e4	1.0	202	1062	1050
PMC-anti	6.73e-10	9.8e4	0.99	957	60	58
turnip	6.67e-10	9.9e4	1.0	176	• 4380	4350
turnip-anti	9.61e-10	9.3e3	0.30	905	○ 1928	179
GS	8.46e-10	62	0.025		3655	2.3
GS-anti	7.97e-10	61	0.025		3730	2.3
			$\lambda = 10^{-10}$			
PMC	1.34e-19	5.0e4	0.71	202	1018	509
PMC-rev	1.34e-19	5.0e4	0.71	202	60	29
turnip-anti	3.01e-20	3.0e4	0.55	905	○ 1694	514
GS	8.24e-20	121	0.035		4899	5.9
GS-anti	8.00e-20	114	0.034		4974	5.7

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.

40×40 lattice graph, $n = 10^4$

algorithm	$\bar{W}_n$	$S_n^2/\bar{W}_n^2$	RE$[\bar{W}_n]$	$\bar{C}$	T(sec)	WNRV
			$\lambda = 10^{-5}$			
PMC	6.1e-27	1.0e4	1	818	2234	2230
turnip-anti	5.2e-35	9988	1	3680	○ 3946	3946
GS	7.98e-10	57	0.076		6183	35
GS-anti	7.88e-10	69	0.083		5980	41
			$\lambda = 10^{-10}$			
PMC	2.0e-134	1.0e4	1	812	2199	2200
turnip-anti	1.9e-33	1.0e4	1	3679	○ 3531	3531
GS	5.0e-20	151	0.12		6034	91
GS-anti	8.9e-20	124	0.11		6688	83

Complete graph with 100 nodes, $n = 10^4$

Gives 4950 links and 5048 shocks.

algorithm	$\bar{W}_n$	$S_n^2/\bar{W}_n^2$	RE$[\bar{W}_n]$	T(sec)	WNRV
		$\lambda = 0.5$			
GS	2.45e-20	109	0.11	3859	42
GS-anti	2.49e-20	128	0.11	4004	51

Extensions

PMC, turnip, and GS could be adapted to rare-event simulation in 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.

Example: Probability that max flow is under a given threshold in a network where links have random capacities. Application to credit risk.

Example: Probability of overflow in a communication network where links have capacities and demand is random.

Etc.

Aproximate zero-variance IS

Suppose the X_j's are independent and $\mathbb{P}[X_j = 0] = q_j$.
We generate $X_1, X_2, \ldots, X_m$ in this order.
Can be seen as a Markov chain with state $\mathbf{Y}_j = (X_1, \ldots, X_j)$ at step j.
Importance sampling (IS) scheme: replace each q_j by $\tilde{q}_j$ and final estimator $1 - \Phi(X_1, \ldots, X_m)$ by

$$\tilde{u} = (1 - \Phi(X_1, \ldots, X_m)) \prod_{j=1}^{m} \left(\frac{q_j}{\tilde{q}_j} \mathbb{I}[X_j = 0] + \frac{1 - q_j}{1 - \tilde{q}_j} \mathbb{I}[X_j = 1] \right).$$

$$\begin{aligned} \widetilde{\mathbb{E}}[\tilde{u}] &= \sum_{\{\mathbf{x}:\Phi(\mathbf{x})=0\}} \prod_{j=1}^{m} \left(\frac{q_j}{\tilde{q}_j} \mathbb{I}[x_j = 0] \tilde{q}_j + \frac{1 - q_j}{1 - \tilde{q}_j} \mathbb{I}[x_j = 1](1 - \tilde{q}_j) \right) \\ &= \sum_{\{\mathbf{x}:\Phi(\mathbf{x})=0\}} \prod_{j=1}^{m} (q_j \mathbb{I}[x_j = 0] + (1 - \tilde{q}_j) \mathbb{I}[x_j = 1]) \; = u. \end{aligned}$$

Challenge: How to choose the $\tilde{q}_j$'s?

Zero-variance scheme: We have

$$\begin{aligned} u_j(x_1,\ldots,x_{j-1}) &\stackrel{\text{def}}{=} \mathbb{P}[\phi(\mathbf{X}) = 0 \mid X_1 = x_1,\ldots,X_{j-1} = x_{j-1}] \\ &= q_j\ u_{j+1}(x_1,\ldots,x_{j-1},0) + (1-q_j)\ u_{j+1}(x_1,\ldots,x_{j-1},1). \end{aligned}$$

Zero-variance scheme: We have

$$\begin{aligned} u_j(x_1,\ldots,x_{j-1}) &\stackrel{\text{def}}{=} \mathbb{P}[\phi(\mathbf{X}) = 0 \mid X_1 = x_1, \ldots, X_{j-1} = x_{j-1}] \\ &= q_j\, u_{j+1}(x_1,\ldots,x_{j-1},0) + (1-q_j)\, u_{j+1}(x_1,\ldots,x_{j-1},1). \end{aligned}$$

Zero-variance importance sampling scheme (ideal): replace q_j with

$$\tilde{q}_j = q_j \frac{u_{j+1}(x_1,\ldots,x_{j-1},0)}{u_j(x_1,\ldots,x_{j-1})}, \quad 1-\tilde{q}_j = (1-q_j)\frac{u_{j+1}(x_1,\ldots,x_{j-1},1)}{u_j(x_1,\ldots,x_{j-1})}.$$

Then the final estimator is always equal to

$$L(X_1,\ldots,X_m) = \prod_{j=1}^{m} \frac{u_j(x_1,\ldots,x_{j-1})}{u_{j+1}(x_1,\ldots,x_j)} = \frac{u_0()}{u_m(x_1,\ldots,x_m)} = u.$$

Practice: replace unknown u_{j+1} by easily-computable approx. $\hat{u}_{j+1}$.

Suppose each $q_j \to 0$ when $\epsilon \to 0$.

Theorem: If

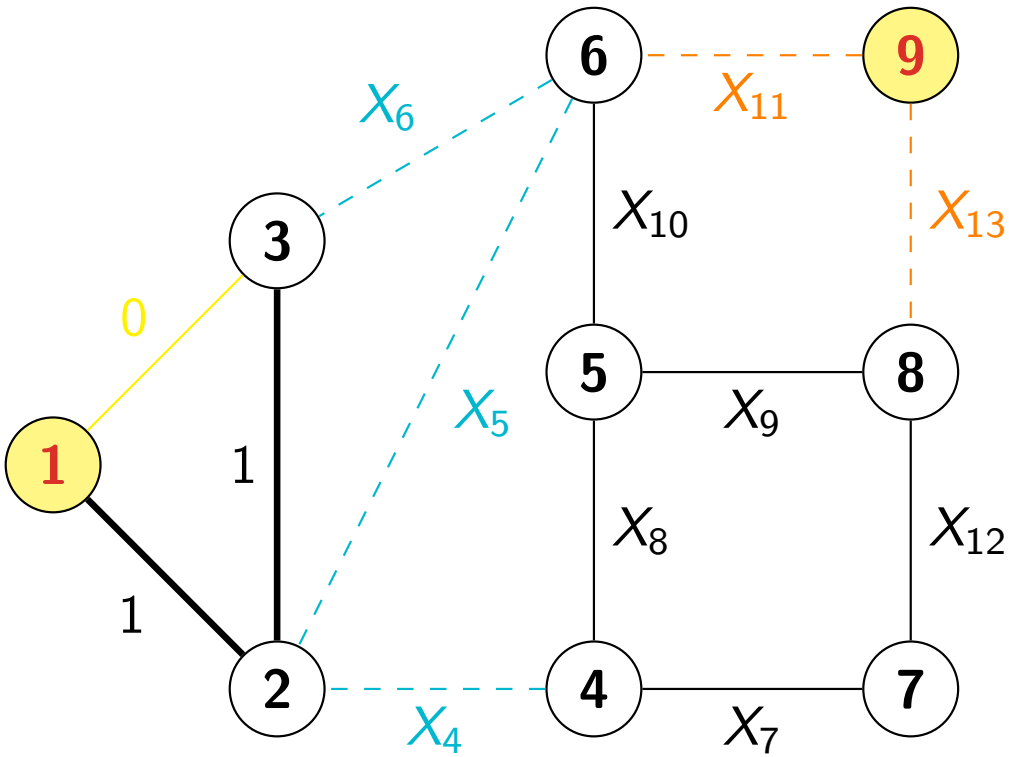
$$\hat{u}_{j+1}(x_1, \ldots, x_j) = a_{j+1}(x_1, \ldots, x_j) u(x_1, \ldots, x_j) + o(u_{j+1}(x_1, \ldots, x_j))$$

with $a_{j+1}(x_1, \ldots, x_j)$ independent of ϵ, then we have BRE.
If $a_{j+1}(x_1, \ldots, x_j) \equiv 1$, then we also have VRE.

Mincut-maxprob approximation of $u_{j+1}(x_1, \ldots, x_j)$.

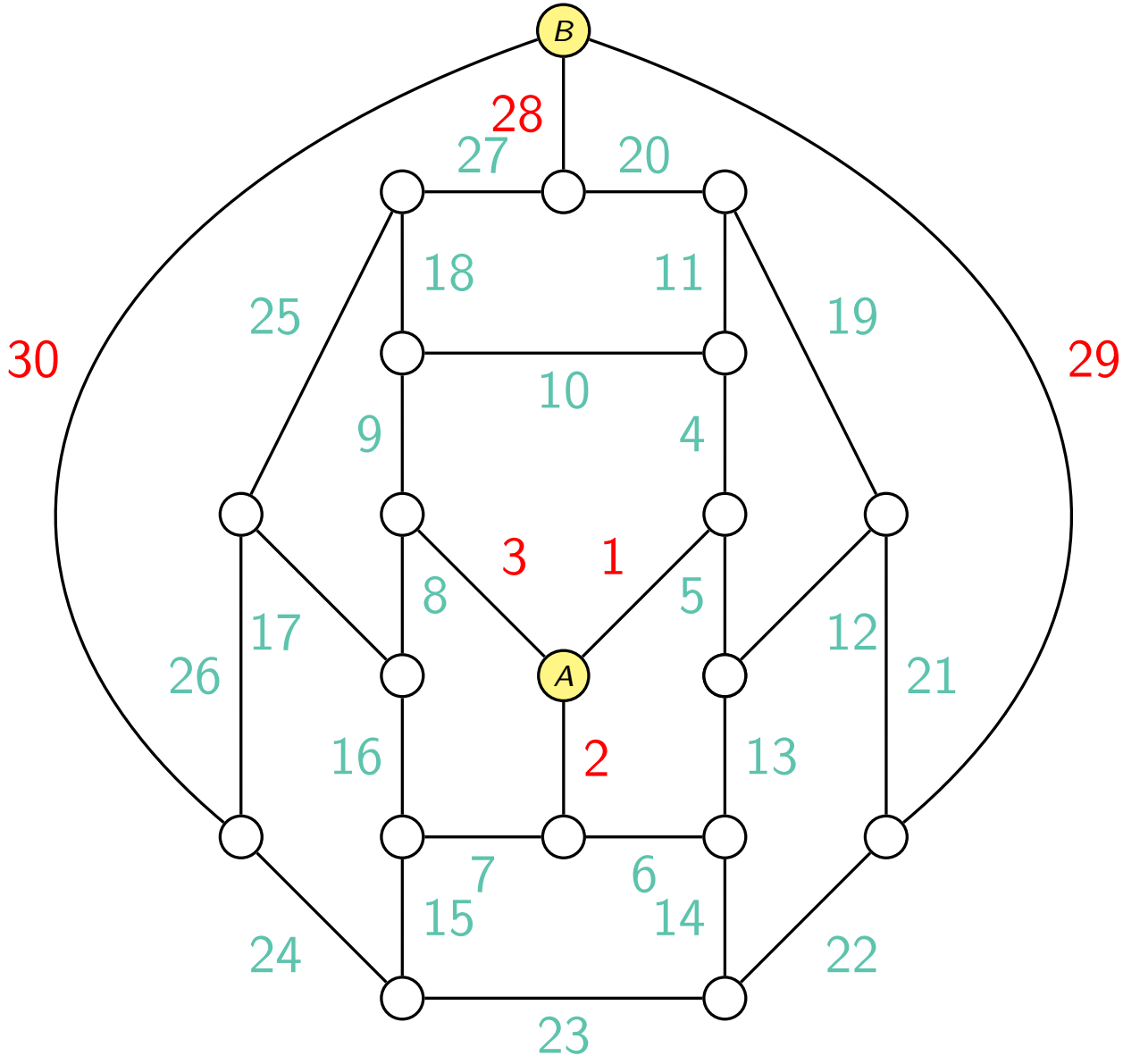
Given $x_1, \ldots, x_j$ fixed, take a minimal cut made with the other edges, that disconnects $\mathcal{V}_0$ and has maximal probability.

Approximate $u_{j+1}(x_1, \ldots, x_j)$ by the probability $\widehat{u}^{\text{mc}}_{j+1}(x_1, \ldots, x_j)$ of this cut, which is the product of its q_j's.



Theorem: The mincut-maxprob approximation always gives BRE.
Under some additional conditions, it also gives VRE.

A dodecahedron network



Results for dodecahedron graph, with all $q_j = \epsilon$, for $n = 10^4$.

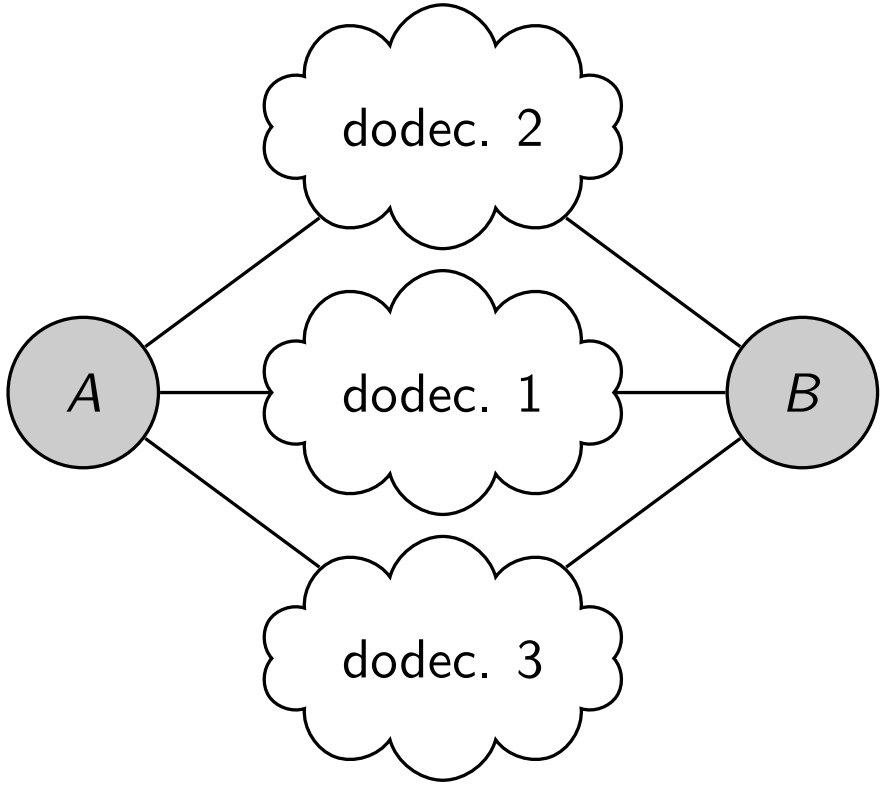
ϵ	estimate	standard dev.	relative error
10^{-1}	2.8960×10^{-3}	3.49×10^{-3}	1.2
10^{-2}	2.0678×10^{-6}	3.42×10^{-7}	0.17
10^{-3}	2.0076×10^{-9}	1.14×10^{-10}	0.057
10^{-4}	2.0007×10^{-12}	3.46×10^{-14}	0.017

Results for dodecahedron graph, with all $q_j = \epsilon$, for $n = 10^4$.

ϵ	estimate	standard dev.	relative error
10^{-1}	2.8960×10^{-3}	3.49×10^{-3}	1.2
10^{-2}	2.0678×10^{-6}	3.42×10^{-7}	0.17
10^{-3}	2.0076×10^{-9}	1.14×10^{-10}	0.057
10^{-4}	2.0007×10^{-12}	3.46×10^{-14}	0.017

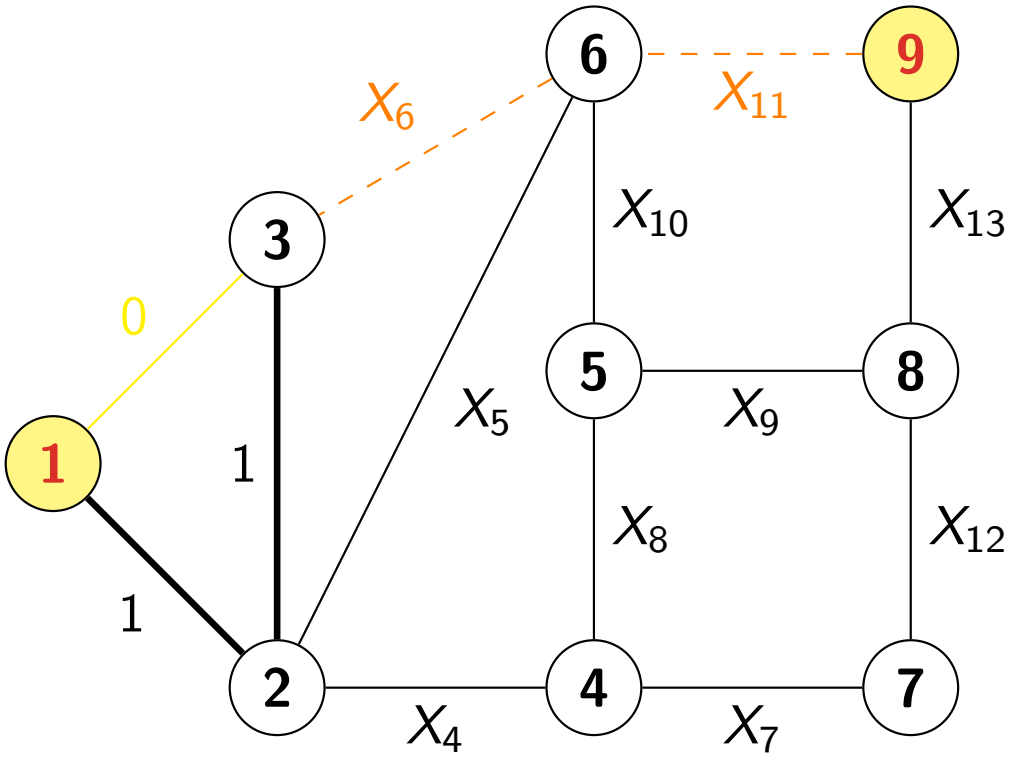
Can combine the method with series-parallel reductions of the graph at each step (WSC 2011 paper).

Three dodecahedron graphs in parallel. $q_j = \epsilon$ and for $n = 10^4$.



ϵ	estimate	standard dev.	relative error
0.10	2.3573×10^{-8}	5.49×10^{-8}	2.3
0.05	2.5732×10^{-11}	3.03×10^{-11}	1.2
0.01	8.7655×10^{-18}	2.60×10^{-18}	0.30

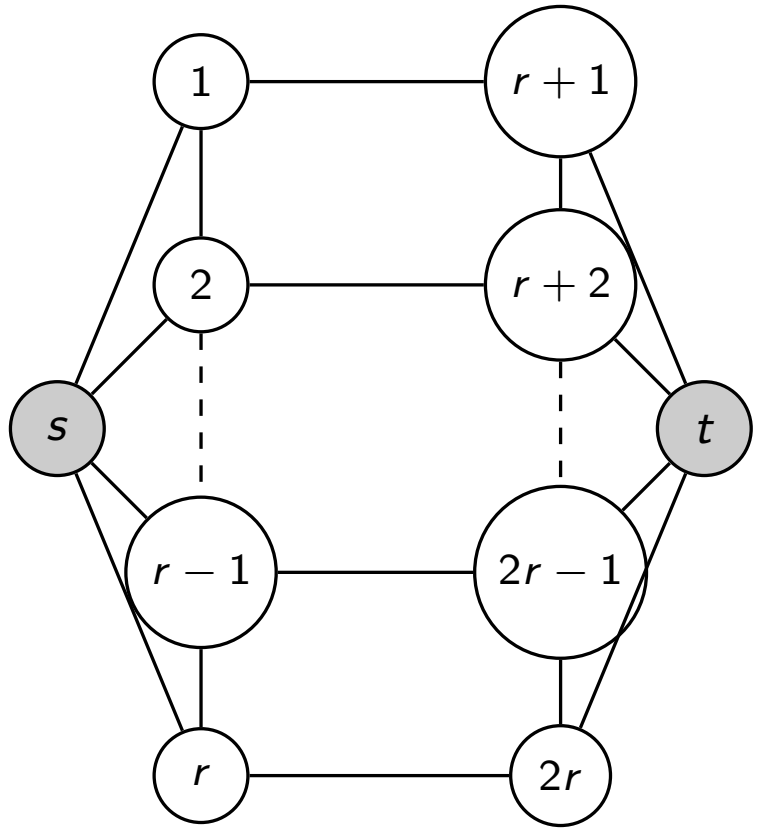
Dual method: **minpath-maxprob approximation of** $u_{j+1}(x_1, \ldots, x_j)$.
Given $x_1, \ldots, x_j$ fixed, take a minimal path made with the other edges, that connects $\mathcal{V}_0$ and has maximal probability.
Approximate $1 - u_{j+1}(x_1, \ldots, x_j)$ by the probability $1 - \widehat{u}^{\text{mc}}_{j+1}(x_1, \ldots, x_j)$ of this path, which is the product of its $(1 - q_i)$'s.



Theorem: The minpath-maxprob approximation always gives BRE for $1 - u$ when the $q_j \to 1$. Under additional conditions, it also gives VRE.

Lemma: $\widehat{u}_{j+1}^{\mathsf{mc}}(x_1, \ldots, x_j) \leq u \leq \widehat{u}_{j+1}^{\mathsf{mp}}(x_1, \ldots, x_j)$.

Example: an $r \times 2$ graph, with $q_j = q$



Original graph has 3 (vertical) mincuts with maximal prob q^r, so $\widehat{u}_1^{\mathrm{mc}}(\emptyset) = q^r$. Also several mincuts of prob q^{r+1}, q^{r+2}, etc.

Several minpaths of length 3, so $\widehat{u}_1^{\mathrm{mp}}(\emptyset) = 1 - (1-q)^3$.

For various r, we selected q so that u is near 10^{-8} in all cases.

Mincut-maxprob approximation:

r	q	$10^8 \hat{u}$	$\widehat{\mathrm{RE}}$	$\hat{u}_1^{\mathrm{mc}}(\emptyset) = q^r$
2	0.00007	1.46	0.33	4.9×10^{-9}
5	0.02	1.06	0.46	3.2×10^{-9}
10	0.1245	1.11	1.8	8.9×10^{-10}
30	0.371	1.14	7.9	1.2×10^{-13}
40	0.427	1.05	9.9	1.6×10^{-15}
50	0.4665	1.08	31	2.7×10^{-17}
70	0.521	1.35	22	1.5×10^{-20}
100	0.575	1.48	40	9.2×10^{-25}
200	0.655	0.48	44	1.8×10^{-37}

Poor behavior for large r. Reason: the several mincuts of prob q^{r+1}, q^{r+2}, etc., contribute significantly and cannot be neglected.

Minpath-maxprob approximation:

r	q	$10^8\,\hat{u}$	$\widehat{\mathrm{RE}}$	$\widehat{u}_1^{\mathsf{mp}}(\emptyset)$
2	0.00007	1.68	66	0.0002
5	0.02	3.18	160	0.058
10	0.1245	1.15	110	0.32
30	0.371	1.36	75	0.75
40	0.427	1.20	36	0.81
50	0.4665	0.98	26	0.84
70	0.521	1.58	17	0.89
90	0.559	1.19	6.6	0.91
100	0.575	1.52	9.8	0.92
200	0.655	1.13	3.9	0.95

Not so good for small r, because the minpaths of prob $(1-q)^4$, etc., contribute significantly.
But much better than mincuts for large r.

A linear combination of two unreliability approximations

We consider an IS scheme that approximates $u_{i+1}(\cdot)$ by the linear combination

$$\widehat{u}_{i+1}(x_1, \ldots, x_i) = \alpha \widehat{u}^{\mathsf{mc}}_{i+1}(x_1, \ldots, x_i) + (1 - \alpha)\widehat{u}^{\mathsf{mp}}_{i+1}(x_1, \ldots, x_i)$$

$\forall i$ and $\forall (x_1, \ldots, x_i) \in \{0, 1\}^i$, for some constant $\alpha \in [0, 1]$ to be chosen.

Want to choose α to minimize the variance $V(\alpha)$ of the IS estimator.

This α can be estimated by stochastic approximation (SA). We find (approximately) a root of $V'(\alpha) \stackrel{\text{def}}{=} (\partial V/\partial \alpha)(\alpha) = 0$.

A linear combination of two unreliability approximations

We consider an IS scheme that approximates $u_{i+1}(\cdot)$ by the linear combination

$$\widehat{u}_{i+1}(x_1, \ldots, x_i) = \alpha \widehat{u}^{\mathrm{mc}}_{i+1}(x_1, \ldots, x_i) + (1 - \alpha)\widehat{u}^{\mathrm{mp}}_{i+1}(x_1, \ldots, x_i)$$

$\forall i$ and $\forall (x_1, \ldots, x_i) \in \{0, 1\}^i$, for some constant $\alpha \in [0, 1]$ to be chosen.

Want to choose α to minimize the variance $V(\alpha)$ of the IS estimator.

This α can be estimated by stochastic approximation (SA). We find (approximately) a root of $V'(\alpha) \stackrel{\mathrm{def}}{=} (\partial V/\partial \alpha)(\alpha) = 0$.

If we were allowed to choose a different $\alpha = \alpha(x_1, \ldots, x_i) \in [0, 1]$ for each $(x_1, \ldots, x_i)$, we could in principle achieve zero variance. But would be too hard to optimize. Choosing a single α is a compromise.

Crude heuristic to estimate α

If we knew u, we could take α for which

$$u = u_1(\emptyset) = \alpha \widehat{u}_1^{\mathsf{mc}}(\emptyset) + (1-\alpha)\widehat{u}_1^{\mathsf{mp}}(\emptyset).$$

That is,

$$\alpha = \frac{\widehat{u}^{\mathsf{mp}}(\emptyset) - u}{\widehat{u}^{\mathsf{mp}}(\emptyset) - \widehat{u}^{\mathsf{mc}}(\emptyset)}.$$

Can replace unknown u in this formula by a rough estimate $\widehat{u}$ obtained from pilot runs.

Learning a good α by stochastic approximation

Start at some $\alpha_0 \in [0, 1]$ and iterate:

$$\alpha_{\ell+1} = \alpha_\ell - \frac{e}{(C+\ell)^\beta} \widehat{V'}(\alpha_\ell),$$

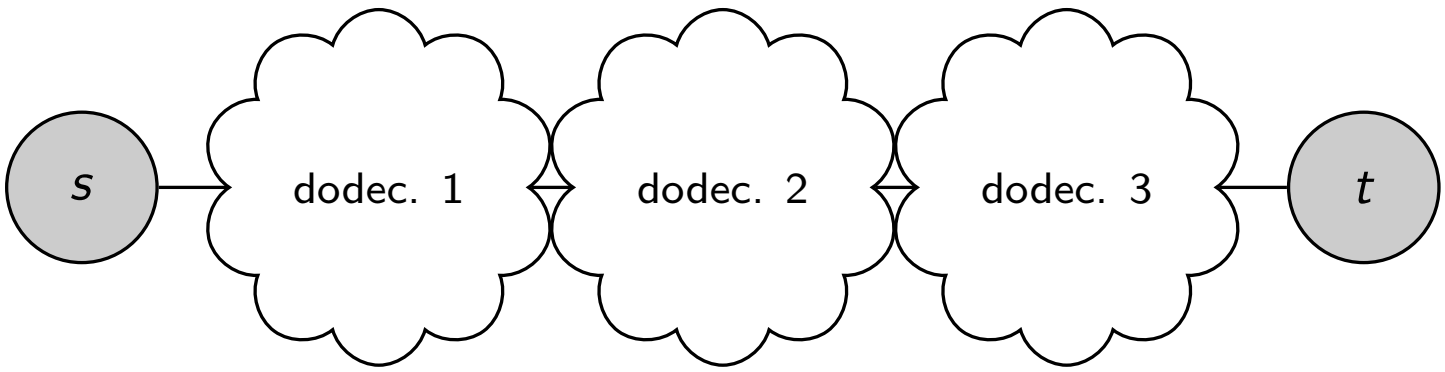
where $\widehat{V'}(\alpha_\ell)$ is an estimate of $V'(\alpha_\ell)$.

An unbiased derivative estimator can be derived by infinitesimal perturbation analysis. Gives a complicated formula but easy to evaluate by simulation.

At the end, take α as the average

$$\bar{\alpha}_{\ell_0,\ell} = \frac{1}{\ell - \ell_0} \sum_{\iota=\ell_0+1}^{\ell} \alpha_\iota.$$

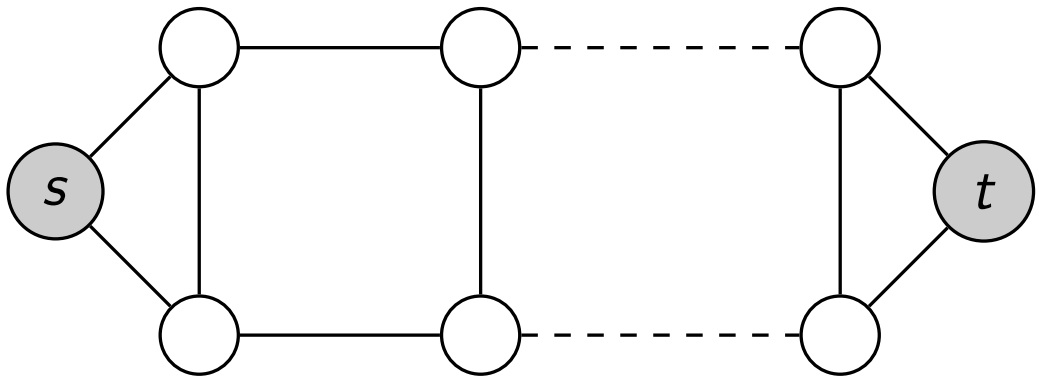
Example: Three dodecahedrons in series



Example: Three dodecahedrons in series

method	q	$\hat{u}$	$\widehat{\mathrm{RE}}$	$\hat{\alpha}$	$\widehat{u}^{\mathrm{mc}}(\emptyset)$	$\widehat{u}^{\mathrm{mp}}(\emptyset)$
MC	10^{-1}	8.577×10^{-3}	2.8			
	10^{-2}	6.173×10^{-6}	1.3			
	10^{-3}	6.012×10^{-9}	1.3			
	10^{-4}	5.989×10^{-12}	1.3			
MP	10^{-1}	8.205×10^{-3}	6.8			
	10^{-2}	4.339×10^{-6}	91			
	10^{-3}	1.002×10^{-9}	0.060			
	10^{-4}	1.000×10^{-12}	0.018			
heuristic	10^{-1}	8.584×10^{-3}	0.75	0.990	10^{-3}	0.794
	10^{-2}	6.015×10^{-6}	0.31	0.9999635	10^{-6}	0.140
	10^{-3}	6.015×10^{-9}	0.28	0.999999665	10^{-9}	1.489×10^{-2}
	10^{-4}	5.997×10^{-12}	0.27	0.999999996	10^{-12}	1.498×10^{-3}
SA	10^{-1}	8.599×10^{-3}	0.71	0.991277		
	10^{-2}	6.188×10^{-6}	0.25	0.999974		
	10^{-3}	6.014×10^{-9}	0.22	0.99999975		
	10^{-4}	5.997×10^{-12}	0.22	0.9999999975		

Two rows of m nodes



Two rows of m nodes

method	q	m	$\hat{u}$	$\widehat{\mathrm{RE}}$	$\hat{\alpha}$	$\widehat{u}^{\mathrm{mc}}(\emptyset)$	$\widehat{u}^{\mathrm{mp}}(\emptyset)$
MC	0.5	2	0.672	0.42			
		10	0.995	2.0			
		20	0.998	2.1			
	0.1	2	0.0329	0.50			
		10	0.123	5.3			
		20	0.518	190			
MP	0.5	2	0.672	0.20			
		10	0.9889	0.049			
		20	0.99982	0.0063			
	0.1	2	0.0329	1.5			
		10	0.1208	1.5			
		20	0.2182	1.3			
heuristic	0.5	2	0.672	0.14	0.33	0.25	0.875
		10	0.988	0.044	0.0255	0.25	0.999
		20	0.99978	0.014	0.02614	0.25	0.9999995
	0.1	2	0.03315	0.56	0.912	10^{-2}	0.270
		10	0.1205	0.39	0.836	10^{-2}	0.686
		20	0.2188	0.34	0.8536	10^{-2}	0.8905
SA	0.5	2	0.672	0.13	0.397		
		10	0.9886	0.045	0.0225		
		20	0.99983	0.006	1.6×10^{-36}		
	0.1	2	0.0330	0.33	0.946		
		10	0.1202	0.34	0.877		
		20	0.2184	0.32	0.828		

References

Z. I. Botev, P. L'Ecuyer, R. Simard, and B. Tuffin, "Static Network Reliability Estimation under the Marshall-Olkin Copula," ACM Transactions on Modeling and Computer Simulation, **26**, 2 (2016), Article 14.

P. L'Ecuyer, G. Rubino, S. Saggadi, and B. Tuffin, "Approximate Zero-Variance Importance Sampling for Static Network Reliability Estimation," IEEE Transactions on Reliability, **8**, 4 (2011), 590–604.

B. Tuffin, S. Saggadi, and P. L'Ecuyer, "An Adaptive Zero-Variance Importance Sampling Approximation for Static Network Dependability Evaluation", Computers and Operations Research, **45** (2014), 51–59.

P. L'Ecuyer, J. Blanchet, B. Tuffin, and P. W. Glynn, "Asymptotic Robustness of Estimators in Rare-Event Simulation", ACM Transactions on Modeling and Computer Simulation, 20, 1 (2010), Article 6, 41 pages.