AUSTRALIAN RESEARCH COUNCIL CENTRE OF EXCELLENCE FOR MATHEMATICAL AND STATISTICAL FRONTIERS

Reliability and Resiliency in Network Infrastructure

Reliability & Monte Carlo

Slava Vaisman

School of Mathematics and Physics

The University of Queensland Australia

`https://www.smp.uq.edu.au/people/RadislavVaisman`

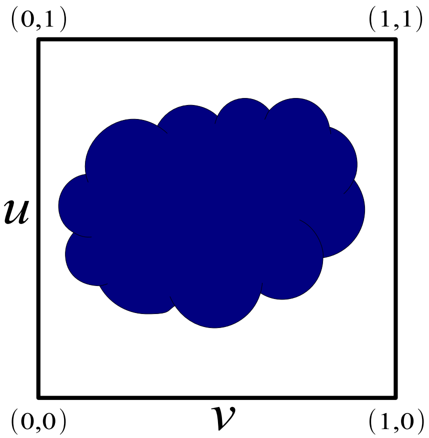
My research

My research lies in the field of applied probability, stochastic simulation and machine learning.

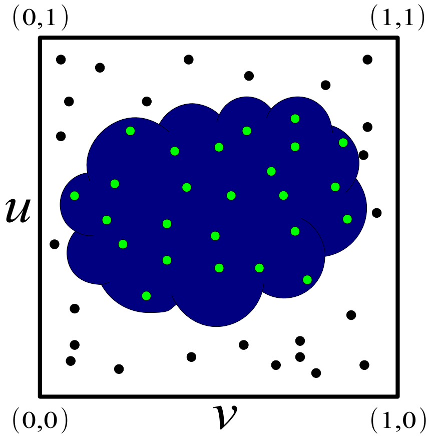
- Advanced Monte Carlo methods for rare event estimation.
- Combinatorial optimization and counting.
- Network reliability.
- Applied Probability.
- Machine learning and data science.
- Design analysis and implementation of algorithms.
- Evolutionary computation.
- Markov Decision Processes and planning under uncertainty.

Regardless if the application domain, my work is mostly concerned with the rare-event setting.

Calculate the blue area



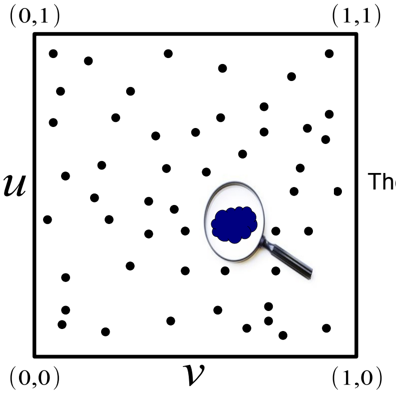
The Monte Carlo Method



Estimator for the blue area

The blue area is approximately equal to $\frac{\text{\# of green points}}{\text{\# of green and black points}}$

Rare events



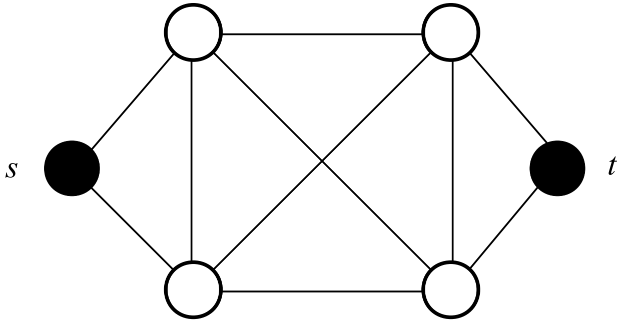
The area of is $\approx 10^{-40}$

- The problem is the high variance of the estimator.
- Most of my research is about finding smarter estimators that avoid the above rare-event setting.

Introduction

Network Reliability

- Given a graph $G = G(V, E)$, a Terminal node set $T \subseteq V$ and an edge failure probability $q_e \ \forall e \in E$, find the probability that the terminal set is connected.
- Many problem's variants are possible.
- In general, the problem belongs to the #P complexity class.



Network reliability with Crude Monte Carlo

We consider a special case of the terminal network reliability problem in which the nodes are completely reliable and all edge failure probabilities are equal; that is, $q_e = q$ for all $e \in E$. Additionally, we assume that the edge failures are independent.

- Generate N independent realizations of G where each edge is *up* with probability $1 - q$.
- Set $X_i = 0$ if the i-th G is $s - t$ connected and $X_i = 1$ otherwise.
- The network unreliability is equal to $\mathbb{E}[X]$, that is, it can be estimated by

$$\frac{1}{N} \sum_{i=1}^{N} X_i.$$

Suppose that $q \ll 1$, that is, the network is highly reliable. In this case we encounter the rare-event setting.

Network' signature

1. Let $e_1, e_2, \ldots, e_m$ be the network edges. Suppose that all of them are initially operational and thus the network is in the *UP* state.
2. Let $\pi = (e_{i_1}, \ldots, e_{i_m})$ be a permutation of edges.
3. Given the permutation $\boldsymbol{\pi}$, start "erasing" edges (change the edges state from *up* to *down*) moving through the permutation from left to right and check the *UP*/*DOWN* state of the network after each step.
4. Find the index j of the first edge e_{i_j}, $j = 1, \ldots, m$ for which the network switches from *UP* to *DOWN*. This index j is called the *anchor* of $\boldsymbol{\pi}$ and is denoted by $a(\boldsymbol{\pi})$.

Next, we assign the uniform distribution on the set of all edge permutations, that is, $\mathbb{P}(\boldsymbol{\Pi} = \boldsymbol{\pi}) = 1/m!$, and define the set

$$A(k) = \{\boldsymbol{\pi} \mid a(\boldsymbol{\pi}) = k\}.$$

Network' signature — the Spectra

Definition (Destruction Spectra)

Let $\mathbf{\Pi}$ be a random permutation and define

$$f_k = \mathbb{P}\left(\mathbf{\Pi} \in A(k)\right) = \frac{|A(k)|}{m!}, \quad k = 0, \ldots, m.$$

Then,

$$S_p = \{f_0, f_1, \ldots, f_m\}$$

is called the *Destruction Spectra*, or simply *D–Spectra* of the network.

Definition (Cumulative D–Spectra)

The cumulative D–Spectra is defined by

$$C_{S_p} = \{F(0), F(1), \ldots, F(m)\},$$

where $F(k) = \sum_{i=0}^{k} f_i, \quad k = 0, \ldots, m.$

The Spectra

- The nice feature of the D–Spectra is that once C_{S_p} is available one can calculate directly the sought network unreliability $\bar{r}(q)$.
- Let us define a *failure set* to be an ordered set of edges such that their failure forces the network to enter the *DOWN* state and denote by N(k)the number of network failure sets of size k.
- Note that each such set is a collection of k edges whose failure results in the *DOWN* state of the network.

It is readily seen that

$$\mathcal{N}(k) = \binom{m}{k} F(k). \tag{1}$$

This statement has a simple combinatorial explanation: $F(k)$ is the fraction of all failure sets of size k among all subsets of size k taken from the set of m components.

Network reliability calculation with Spectra

Moreover, note that the following holds.

- The network is *DOWN* if and only if it is in one of its failure sets.
- For fixed q each failure set of size k has the probability $q^k(1-q)^{m-k}$.

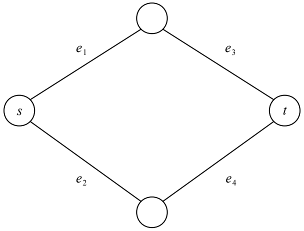
Combining this with (1) we obtain

$$\begin{aligned}\overline{r}(q) &= \sum_{k=0}^{m} \binom{m}{k} F(k) q^k (1-q)^{m-k} \\ &= \sum_{k=0}^{m} \mathcal{N}(k) q^k (1-q)^{m-k}.\end{aligned}$$

With this equation, it only remains to calculate the D–Spectra in order to calculate the network reliability for any q.

Network reliability calculation with Spectra example

As an example, consider the simple graph in the Figure and suppose that $K = \{s, t\}$.

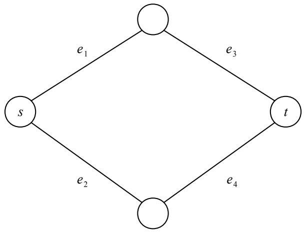


- It is clear that zero or one edge removal cannot bring the network to the *DOWN* state so $f_0 = f_1 = 0$.
- In order to calculate f_2, f_3 and f_4, consider the following Table.

Network reliability calculation with Spectra example

Table: Permutation–anchor.

π	$a(\pi)$	π	$a(\pi)$	π	$a(\pi)$
(e_1, e_2, e_3, e_4)	2	(e_2, e_3, e_4, e_1)	2	(e_3, e_4, e_2, e_1)	2
(e_1, e_2, e_4, e_3)	2	(e_2, e_3, e_1, e_4)	2	(e_3, e_4, e_1, e_2)	2
(e_1, e_3, e_2, e_4)	3	(e_2, e_4, e_3, e_1)	3	(e_4, e_1, e_2, e_3)	2
(e_1, e_3, e_4, e_2)	3	(e_2, e_4, e_1, e_3)	3	(e_4, e_1, e_3, e_2)	2
(e_1, e_4, e_2, e_3)	2	(e_3, e_1, e_2, e_4)	3	(e_4, e_2, e_3, e_1)	3
(e_1, e_4, e_3, e_2)	2	(e_3, e_1, e_4, e_2)	3	(e_4, e_2, e_1, e_3)	3
(e_2, e_1, e_4, e_3)	2	(e_3, e_2, e_4, e_1)	2	(e_4, e_3, e_2, e_1)	2
(e_2, e_1, e_3, e_4)	2	(e_3, e_2, e_1, e_4)	2	(e_4, e_3, e_1, e_2)	2



Network reliability calculation with Spectra example

From the data presented in the Table, the D–Spectra is given by

$$f_0 = f_1 = 0,\ f_2 = 16/24,\ f_3 = 8/24 \text{ and } f_4 = 0,$$

so we arrive at

$$S_p = \{0, 0, 2/3, 1/3, 0\} \text{ and } C_{S_p} = \{0, 0, 2/3, 1, 1\}.$$

As soon as we obtain this Spectra the network unreliability can be readily calculated for any q via

$$\overline{r}\,(q) = \sum_{k=0}^{m} \binom{m}{k} F(k) q^k (1-q)^{m-k} = \sum_{k=0}^{m} \mathcal{N}(k) q^k (1-q)^{m-k}.$$

Note that the spectra does not depend on q. That is. calculate the Spectra once, and get the unreliability for any q!

The construction Spectra

- Sometimes, it is more convenient to work with an equivalent Spectra object called the *Construction Spectra*.
- Under the construction settings, we start with all edges being in state *down*. In this case the network is clearly in the *DOWN* state too.
- The construction and the destruction spectra are equivalent.
- Specifically, we have the construction spectra and the commutative construction spectra.
- For the reliability calculation, we will use the a formula which is similar to the one used in the destruction spectra case.

Unfortunately, both the construction and the destruction Spectra are generally not available analytically and as consequence a Monte Carlo procedure should be applied. One of the widely adopted approaches, the Permutation Monte Carlo (PMC).

The PMC Algorithm

Given a graph $G(V, E, K)$ such that $|E| = m$ and a sample size N, execute the following steps.

1. **Step 1 (Initialization):** Set

$$\widehat{S_p} \equiv \{\widehat{f_0}, \ldots, \widehat{f_m}\} \longleftarrow \underbrace{\{0, 0, \ldots, 0\}}_{m+1}.$$

2. **Step 2 (Main loop):** Repeat N times.
   - 2.1 $\mathbf{\Pi} \leftarrow (e_{i_1}, \ldots, e_{i_m})$ (generate a random edge permutation).
   - 2.2 $k \leftarrow a(\mathbf{\Pi})$ (find the anchor).
   - 2.3 $\widehat{f_k} \longleftarrow \widehat{f_k} + 1$.
3. **Step 3 (Calculate D–Spectra):**

$$\widehat{f_k} \longleftarrow \frac{\widehat{f_k}}{N} \quad \text{for } k = 0, \ldots, m.$$

4. **Step 4 (Calculate Cumulative D–Spectra):**

$$\widehat{F(k)} \longleftarrow \sum_{i=0}^{k} \widehat{f_i} \quad \text{for } k = 0, \ldots, m.$$

Is Spectra actually a silver bullet?

Let us turn our attention to the Spectra estimation under rare–event settings and consider an example Spectra in the following Table.

C_{S_p}				$\widehat{C_{S_p}}$			
$F(0)$	0	$F(10)$	0.3	$\widehat{F(0)}$	0	$\widehat{F(10)}$	0.3
$F(1)$	0	$F(11)$	0.5	$\widehat{F(1)}$	0	$\widehat{F(11)}$	0.5
$F(2)$	0	$F(12)$	0.7	$\widehat{F(2)}$	0	$\widehat{F(12)}$	0.7
$F(3)$	0	$F(13)$	0.9	$\widehat{F(3)}$	0	$\widehat{F(13)}$	0.9
$F(4)$	10^{-12}	$F(14)$	1	$\widehat{F(4)}$	0	$\widehat{F(14)}$	1
$F(5)$	10^{-10}	$F(15)$	1	$\widehat{F(5)}$	0	$\widehat{F(15)}$	1
$F(6)$	10^{-8}	$F(16)$	1	$\widehat{F(6)}$	0	$\widehat{F(16)}$	1
$F(7)$	10^{-6}	$F(17)$	1	$\widehat{F(7)}$	10^{-6}	$\widehat{F(17)}$	1
$F(8)$	10^{-3}	$F(18)$	1	$\widehat{F(8)}$	10^{-3}	$\widehat{F(18)}$	1
$F(9)$	0.1	$F(19)$	1	$\widehat{F(9)}$	0.1	$\widehat{F(19)}$	1

The left part of the table presents the "real" Spectra values (C_{S_p}) and the right stands for their (fake) "estimates" ($\widehat{C_{S_p}}$). Suppose that $\widehat{C_{S_p}}$ was obtained using the PMC Algorithm.

We deliberately set the "estimates" to be equal to the real values in order to stress the importance of the rare components ($F(4)$, $F(5)$ and $F(6)$)

Is Spectra actually a silver bullet?

- Note that the proposed PMC Algorithm will not be able to estimate the left part of the Spectra ($F(4), F(5)$ and $F(6)$) in a reliable manner using any manageable sample size, say $N = 10^7$.
- This problem is due to the well-known issue of Crude Monte Carlo algorithms under rare–event settings. For example, consider the value of f_4. Having in mind that $\mathbb{P}(a(\boldsymbol{\pi}) = 4) = f_4 = 10^{-12}$, the Relative Error (RE) of the PMC Algorithm for the $\widehat{f_4}$ is given by

$$\mathrm{RE} = \sqrt{\frac{\mathrm{Var}\left(\widehat{f_4}\right)}{\mathbb{E}\left(\widehat{f_4}\right)^2 N}} = \sqrt{\frac{f_4(1-f_4)}{f_4^2 N}}$$
$$= \sqrt{\frac{10^{-12}\cdot(1-10^{-12})}{10^{-(12\cdot 2)}N}} \approx 10^6/\sqrt{N},$$

where N is the sample size. The above equation suggests that in order to achieve even a modest (say 10%) RE, the value of N should be huge.

Is Spectra actually a silver bullet?

- Unfortunately, these small Spectra values are important.
- Calculating the unreliability using the true and the estimated values of Spectra values, we get the following results. Here $\overline{r}(q)$ and $\widehat{\overline{r}(q)}$ using the exact and the approximated Spectra respectively.

Table: Network *DOWN* probabilities for different values of q.

q	$\overline{r}(q)$	$\widehat{\overline{r}(q)}$
10^{-10}	$3.88 \cdot 10^{-49}$	$5.04 \cdot 10^{-72}$
10^{-9}	$3.88 \cdot 10^{-45}$	$5.04 \cdot 10^{-65}$
10^{-8}	$3.88 \cdot 10^{-41}$	$5.04 \cdot 10^{-58}$
10^{-7}	$3.88 \cdot 10^{-37}$	$5.04 \cdot 10^{-51}$
10^{-6}	$3.88 \cdot 10^{-33}$	$5.05 \cdot 10^{-44}$
10^{-5}	$3.89 \cdot 10^{-29}$	$5.11 \cdot 10^{-37}$
10^{-4}	$3.99 \cdot 10^{-25}$	$5.00 \cdot 10^{-30}$
10^{-3}	$5.37 \cdot 10^{-21}$	$1.34 \cdot 10^{-22}$
10^{-2}	$1.62 \cdot 10^{-14}$	$1.58 \cdot 10^{-14}$
10^{-1}	$4.71 \cdot 10^{-6}$	$4.71 \cdot 10^{-6}$

Is Spectra actually a silver bullet?

- One can easily note that the right column of the Table is a clear underestimation for $q \leqslant 10^{-4}$.
- A more careful observation of the results reveals the importance of the leftmost values of Spectra. Note that $\widehat{F(4)}, \widehat{F(5)}$ and $\widehat{F(6)}$ were "evaluated" to zero.
- We shall summarize the discussion as follows.

For highly reliable networks (that is, networks with small edge failure probabilities q), the PMC Algorithm fails to produce reliable results because it requires calculations that involve the accurate estimation of rare–event probabilities.

Research question

Can we show a class of networks that does not have a rare-event problem?

My feeling...

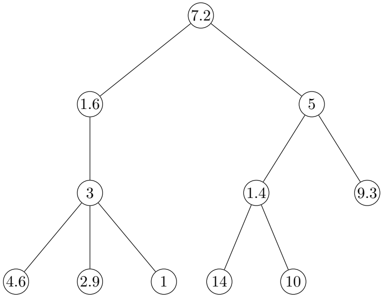
- Consider a class of random networks. (What class?)
- Show that on average, there are no rare-event spectra components.
- Show that this happens with high probability.
- Result — PMC algorithm works with high probability.

Part I

Advanced MC Methods for Network Reliability estimation

The Stochastic Enumeration Method

The Tree Counting Problem Definition



- Consider a rooted tree $T = (\mathcal{V}, \mathcal{E})$ with node set $\mathcal{V}$ and edge set $\mathcal{E}$.
- Which each node v is associated a cost $c(v) \in \mathbb{R}$, (it is also possible that $C(v)$ is a random variable).
- The main quantity of interest is the total cost of the tree,

$$\mathrm{Cost}(T) = \sum_{v \in \mathcal{V}} c(v), \text{ or for r.v:}$$

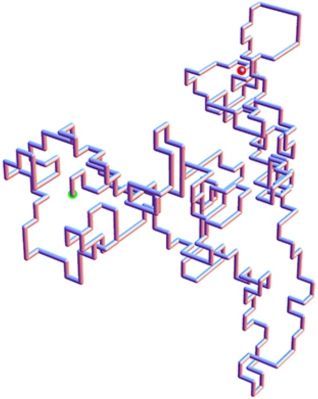
$$\mathrm{Cost}(T) = \mathbb{E}\left(\sum_{v \in \mathcal{V}} C(v)\right).$$

- Linear time solution? (BFS, DFS).

What if the set $|\mathcal{V}|$ is large? Think about querying a large hierarchical structure (database).

Is this an interesting problem? (Self-avoiding walk)

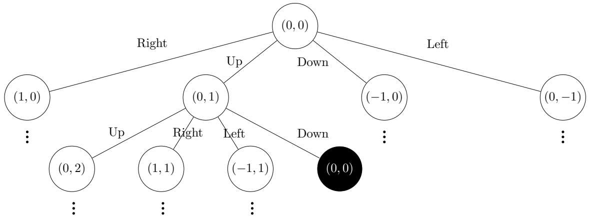
Example 1: Self-avoiding walk — models the real-life behavior of chain-like entities such as polymers.



Question: How many Self-avoiding walks of length *n* exist?

Modelling the Self-avoiding walk as a tree counting problem

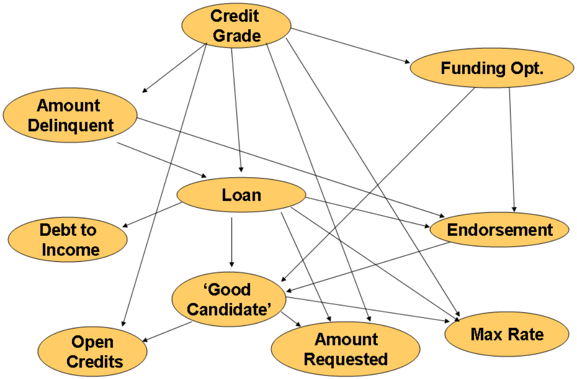
- Consider a Self-avoiding walk in 2D.
- Start from the origin: $(0, 0)$ is the tree root node.
- Each node edge corresponds to a direction {Right, Left, Up Down}.



- Note that each path (of length n and without black nodes), from the tree root to a leaf corresponds to a valid self-avoiding walk.
- Setting $c(v) = 1$ to each such leaf, (and $c(v) = 0$ otherwise), completes the reduction.

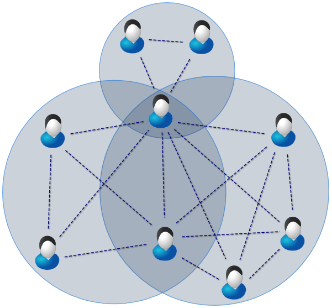
Is this an interesting problem? (Inference)

Example 2: Probabilistic inference in Bayesian networks.



Is this an interesting problem? (Social network analysis)

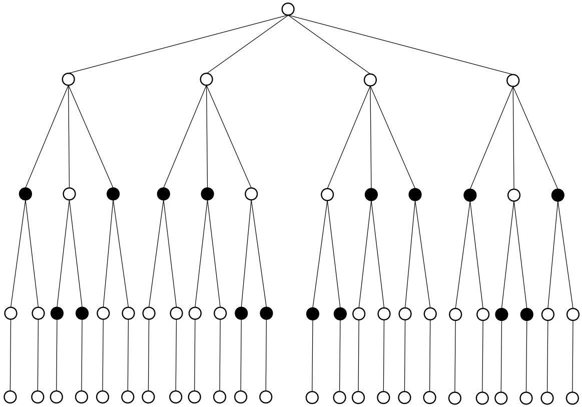
Example 3: Cliques in a graph — counting cliques in (social) network.
Q: How many cliques of size k exist?



A group of individuals that share the same interests can be modelled by a corresponding clique in a graph. Can we target such groups, say via an advertisement or other social activity?

Estimating the Spectra

Consider the set of all edge permutations. With these permutations it is possible to build a tree object that we call the *permutation tree*, for which each path from the root to a leaf corresponds to some specific permutation π.



Estimating the Spectra

The exact edge mapping from the leftmost to the rightmost permutation in the above Figure is as follows.

$$\begin{gathered}\{\{e_1,e_2,e_3,e_4\},\{e_1,e_2,e_4,e_3\},\{e_1,e_3,e_2,e_4\},\{e_1,e_3,e_4,e_2\},\\ \{e_1,e_4,e_2,e_3\},\{e_1,e_4,e_3,e_2\},\{e_2,e_1,e_3,e_4\},\{e_2,e_1,e_4,e_3\},\\ \{e_2,e_3,e_1,e_4\},\{e_2,e_3,e_4,e_1\},\{e_2,e_4,e_1,e_3\},\{e_2,e_4,e_3,e_1\},\\ \{e_3,e_1,e_2,e_4\},\{e_3,e_1,e_4,e_2\},\{e_3,e_2,e_1,e_4\},\{e_3,e_2,e_4,e_1\},\\ \{e_3,e_4,e_1,e_2\},\{e_3,e_4,e_2,e_1\},\{e_4,e_1,e_2,e_3\},\{e_4,e_1,e_3,e_2\},\\ \{e_4,e_2,e_1,e_3\},\{e_4,e_2,e_3,e_1\},\{e_4,e_3,e_2,e_1\},\{e_4,e_3,e_1,e_2\}\}\end{gathered}$$

- By definition, each path from the root to a leaf corresponds to a unique edge permutation π and each black vertex corresponds to this permutation's anchor, $a(\pi)$.
- We can now count the anchor vertices at each level of the tree (by setting say a cost of 1 to each black vertex and zero otherwise), and recover the Spectra object.

Is the tree counting problem is an interesting problem? (Graph polynomials)

1. Reliability polynomial.
2. Tutte polynomial.
3. Independence polynomial.
4. Chromatic polynomial.
5. ...
6. ...
7. ...

Applications

Combinatorics, Statistical Physics and Network Reliability.

Is this an interesting problem? (Optimization)

While working for industry, I used a similar tree counting method for optimization of mobile all-weather air defence system.

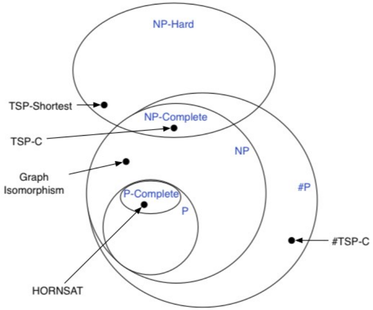
A short summary

1. Many interesting problems from different research fields can be modeled by the tree counting problem.
2. The reduction is relatively straight-forward in many cases.

Consequentially, a general tool for handling the tree counting problem can be very useful.

Is this hard?

The general problem of estimating the cost of a tree is in #P, (Valiant, 1979).



Bad news

An existence of "computationally efficient approximation algorithm" will imply P=NP.

Previous work

Donald E. Knuth (1975)
Estimating the Efficiency of Backtrack Programs
Math. Comp. 29.

Paul W. Purdom (1978)
Tree Size by Partial Backtracking
SIAM J. Comput. 7(4) 481-491.

Pang C. Chen (1992)
Heuristic Sampling: A Method for Predicting the Performance of Tree Searching Programs.
SIAM J. Comput. 21(2) 295-315.

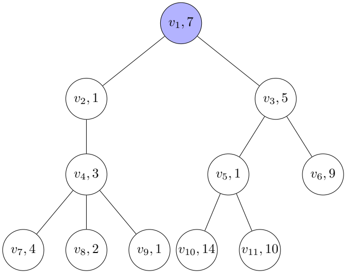
Few additional attempts – based on Knuth's estimator.

Knuth's estimator

Input: A tree T_v of height h, rooted at v.
Output: An unbiased estimator C of the total cost of tree T_v.

1. **(Initialization):** Set $k \leftarrow 0$, $D \leftarrow 1$, $X_0 = v$ and $C \leftarrow c(X_0)$. Here D is the product of all node degrees encountered in the tree.
2. **(Compute the successors):** Let $S(X_k)$ be the set of all successors of X_k and let D_k be the number of elements of $S(X_k)$. If $k = h$ or when $S(X_k)$ is empty, set $D_k = 0$.
3. **(Terminal position?):** If $D_k = 0$, the algorithm stops, returning C as an estimator of $\mathrm{Cost}(T_v)$.
4. **(Advance):** Choose an element $X_{k+1} \in S(X_k)$ at random, each element being equally likely. (Thus, each choice occurs with probability $1/D_k$.) Set $D \leftarrow D_k D$, then set $C \leftarrow C + c(X_{k+1})D$. Increase k by 1 and return to Step 2.

$$k = 0,\ D = 1,\ X_0 = v_1,\ C = 7.$$

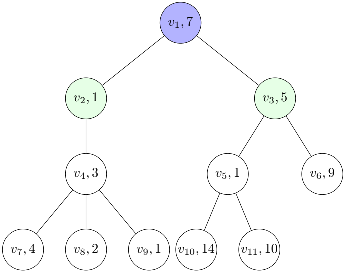


Knuth's estimator

Input: A tree T_v of height h, rooted at v.
Output: An unbiased estimator C of the total cost of tree T_v.

1. **(Initialization):** Set $k \leftarrow 0$, $D \leftarrow 1$, $X_0 = v$ and $C \leftarrow c(X_0)$. Here D is the product of all node degrees encountered in the tree.
2. **(Compute the successors):** Let $S(X_k)$ be the set of all successors of X_k and let D_k be the number of elements of $S(X_k)$. If $k = h$ or when $S(X_k)$ is empty, set $D_k = 0$.
3. **(Terminal position?):** If $D_k = 0$, the algorithm stops, returning C as an estimator of $\mathrm{Cost}(T_v)$.
4. **(Advance):** Choose an element $X_{k+1} \in S(X_k)$ at random, each element being equally likely. (Thus, each choice occurs with probability $1/D_k$.) Set $D \leftarrow D_k D$, then set $C \leftarrow C + c(X_{k+1})D$. Increase k by 1 and return to Step 2.

$$S(X_0) = \{v_2, v_3\},\ D_0 = 2.$$

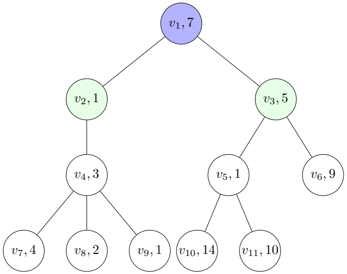


Knuth's estimator

Input: A tree T_v of height h, rooted at v.
Output: An unbiased estimator C of the total cost of tree T_v.

1. **(Initialization):** Set $k \leftarrow 0$, $D \leftarrow 1$, $X_0 = v$ and $C \leftarrow c(X_0)$. Here D is the product of all node degrees encountered in the tree.
2. **(Compute the successors):** Let $S(X_k)$ be the set of all successors of X_k and let D_k be the number of elements of $S(X_k)$. If $k = h$ or when $S(X_k)$ is empty, set $D_k = 0$.
3. **(Terminal position?):** If $D_k = 0$, the algorithm stops, returning C as an estimator of $\mathrm{Cost}(T_v)$.
4. **(Advance):** Choose an element $X_{k+1} \in S(X_k)$ at random, each element being equally likely. (Thus, each choice occurs with probability $1/D_k$.) Set $D \leftarrow D_k D$, then set $C \leftarrow C + c(X_{k+1})D$. Increase k by 1 and return to Step 2.

$$S(X_0) = \{v_2, v_3\},\ D_0 = 2.$$

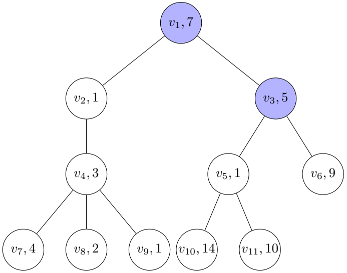


Knuth's estimator

Input: A tree T_v of height h, rooted at v.
Output: An unbiased estimator C of the total cost of tree T_v.

1. **(Initialization):** Set $k \leftarrow 0$, $D \leftarrow 1$, $X_0 = v$ and $C \leftarrow c(X_0)$. Here D is the product of all node degrees encountered in the tree.
2. **(Compute the successors):** Let $S(X_k)$ be the set of all successors of X_k and let D_k be the number of elements of $S(X_k)$. If $k = h$ or when $S(X_k)$ is empty, set $D_k = 0$.
3. **(Terminal position?):** If $D_k = 0$, the algorithm stops, returning C as an estimator of $\mathrm{Cost}(T_v)$.
4. **(Advance):** Choose an element $X_{k+1} \in S(X_k)$ at random, each element being equally likely. (Thus, each choice occurs with probability $1/D_k$.) Set $D \leftarrow D_k D$, then set $C \leftarrow C + c(X_{k+1})D$. Increase k by 1 and return to Step 2.

$$k = 1,\ X_1 = v_3,\ D = 1 \cdot D_0 = 2,$$
$$C = 7 + 5 \cdot 2 = 17.$$

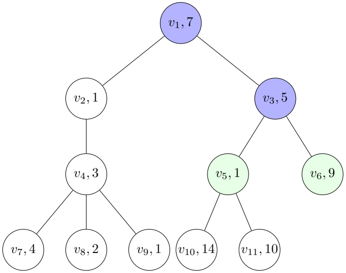


Knuth's estimator

Input: A tree T_v of height h, rooted at v.
Output: An unbiased estimator C of the total cost of tree T_v.

1. **(Initialization):** Set $k \leftarrow 0$, $D \leftarrow 1$, $X_0 = v$ and $C \leftarrow c(X_0)$. Here D is the product of all node degrees encountered in the tree.
2. **(Compute the successors):** Let $S(X_k)$ be the set of all successors of X_k and let D_k be the number of elements of $S(X_k)$. If $k = h$ or when $S(X_k)$ is empty, set $D_k = 0$.
3. **(Terminal position?):** If $D_k = 0$, the algorithm stops, returning C as an estimator of $\mathrm{Cost}(T_v)$.
4. **(Advance):** Choose an element $X_{k+1} \in S(X_k)$ at random, each element being equally likely. (Thus, each choice occurs with probability $1/D_k$.) Set $D \leftarrow D_k D$, then set $C \leftarrow C + c(X_{k+1})D$. Increase k by 1 and return to Step 2.

$$S(X_1) = \{v_5, v_6\},\ D_1 = 2.$$

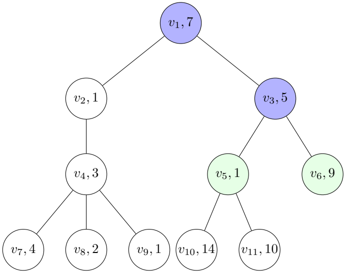


Knuth's estimator

Input: A tree T_v of height h, rooted at v.
Output: An unbiased estimator C of the total cost of tree T_v.

1. **(Initialization):** Set $k \leftarrow 0$, $D \leftarrow 1$, $X_0 = v$ and $C \leftarrow c(X_0)$. Here D is the product of all node degrees encountered in the tree.
2. **(Compute the successors):** Let $S(X_k)$ be the set of all successors of X_k and let D_k be the number of elements of $S(X_k)$. If $k = h$ or when $S(X_k)$ is empty, set $D_k = 0$.
3. **(Terminal position?):** If $D_k = 0$, the algorithm stops, returning C as an estimator of $\mathrm{Cost}(T_v)$.
4. **(Advance):** Choose an element $X_{k+1} \in S(X_k)$ at random, each element being equally likely. (Thus, each choice occurs with probability $1/D_k$.) Set $D \leftarrow D_k D$, then set $C \leftarrow C + c(X_{k+1})D$. Increase k by 1 and return to Step 2.

$$S(X_1) = \{v_5, v_6\},\ D_1 = 2.$$



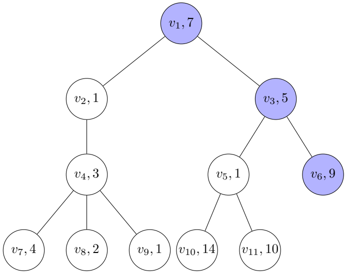
Knuth's estimator

Input: A tree T_v of height h, rooted at v.
Output: An unbiased estimator C of the total cost of tree T_v.

1. **(Initialization):** Set $k \leftarrow 0$, $D \leftarrow 1$, $X_0 = v$ and $C \leftarrow c(X_0)$. Here D is the product of all node degrees encountered in the tree.
2. **(Compute the successors):** Let $S(X_k)$ be the set of all successors of X_k and let D_k be the number of elements of $S(X_k)$. If $k = h$ or when $S(X_k)$ is empty, set $D_k = 0$.
3. **(Terminal position?):** If $D_k = 0$, the algorithm stops, returning C as an estimator of $\mathrm{Cost}(T_v)$.
4. **(Advance):** Choose an element $X_{k+1} \in S(X_k)$ at random, each element being equally likely. (Thus, each choice occurs with probability $1/D_k$.) Set $D \leftarrow D_k D$, then set $C \leftarrow C + c(X_{k+1})D$. Increase k by 1 and return to Step 2.

$$k = 2, \; X_2 = v_6, \; D = 2 \cdot D_1 = 4,$$

$$C = 7 + 5 \cdot 2 + 9 \cdot 4 = 53.$$

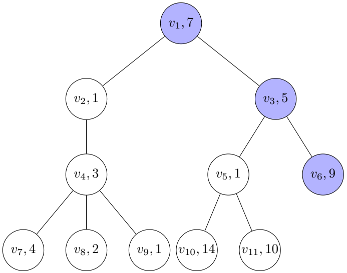


Knuth's estimator

Input: A tree T_v of height h, rooted at v.
Output: An unbiased estimator C of the total cost of tree T_v.

1. **(Initialization):** Set $k \leftarrow 0$, $D \leftarrow 1$, $X_0 = v$ and $C \leftarrow c(X_0)$. Here D is the product of all node degrees encountered in the tree.
2. **(Compute the successors):** Let $S(X_k)$ be the set of all successors of X_k and let D_k be the number of elements of $S(X_k)$. If $k = h$ or when $S(X_k)$ is empty, set $D_k = 0$.
3. **(Terminal position?):** If $D_k = 0$, the algorithm stops, returning C as an estimator of $\mathrm{Cost}(T_v)$.
4. **(Advance):** Choose an element $X_{k+1} \in S(X_k)$ at random, each element being equally likely. (Thus, each choice occurs with probability $1/D_k$.) Set $D \leftarrow D_k D$, then set $C \leftarrow C + c(X_{k+1})D$. Increase k by 1 and return to Step 2.

$$X_2 = v_6,\ S(X_2) = \emptyset,\ D_2 = 0.$$



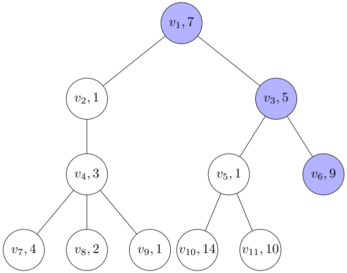
Knuth's estimator

Input: A tree T_v of height h, rooted at v.
Output: An unbiased estimator C of the total cost of tree T_v.

1. **(Initialization):** Set $k \leftarrow 0$, $D \leftarrow 1$, $X_0 = v$ and $C \leftarrow c(X_0)$. Here D is the product of all node degrees encountered in the tree.
2. **(Compute the successors):** Let $S(X_k)$ be the set of all successors of X_k and let D_k be the number of elements of $S(X_k)$. If $k = h$ or when $S(X_k)$ is empty, set $D_k = 0$.
3. **(Terminal position?):** If $D_k = 0$, the algorithm stops, returning C as an estimator of $\mathrm{Cost}(T_v)$.
4. **(Advance):** Choose an element $X_{k+1} \in S(X_k)$ at random, each element being equally likely. (Thus, each choice occurs with probability $1/D_k$.) Set $D \leftarrow D_k D$, then set $C \leftarrow C + c(X_{k+1})D$. Increase k by 1 and return to Step 2.

$C = 53.$

Reached terminal node. Note that $\mathrm{Cost}(T) = 57$.



A very important example!

We are going to discuss a crucial issue, that occurs when applying a Monte Carlo algorithm.

Efficiency of randomized algorithms

- Let X_i be a random variable with finite expected value μ and finite non-zero variance σ^2.
- Define:
$$\overline{X} = \frac{1}{N}\sum_{i=1}^{N} X_i.$$
- From the Chebyshev's inequality:
$$\mathbb{P}\left(|\overline{X} - \mu| > \varepsilon\mu\right) \leqslant \frac{\sigma^2}{N\mu^2\varepsilon^2} \leqslant \delta,$$
for some predefined ε and δ. For example, $\varepsilon = 0.05$, and $\delta = 0.03$, corresponds to the statement that $\overline{X}$ is within 5% interval from the true value, with probability at least 97%.
- This requires the sample size to be $N = \frac{\sigma^2}{\delta\mu^2\varepsilon^2}$, thus, N is proportional to $\mathrm{CV}^2 = \sigma^2/\mu^2$ — the so called squared coefficient of variation.

Is Knuth's algorithm always work? (Rare-events)

Consider the "hair brush" tree T and suppose that the costs of all vertices are zero except for v_{n+1}, which has a cost of unity.

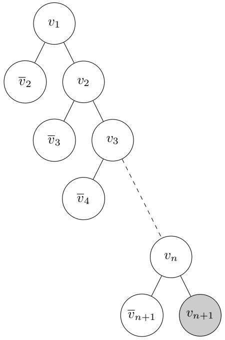


Figure: The hair brush tree.

The expectation and variance of the Knuth's estimator are

$$\mathbb{E}\left(C\right) = \frac{1}{2^n} \cdot 2^n \cdot 1 + \frac{2^n - 1}{2^n} \cdot D' \cdot 0 = 1,$$

and

$$\mathbb{E}\left(C^2\right) = \frac{1}{2^n} \cdot \left(2^n \cdot 1\right)^2 + \\ + \frac{2^n - 1}{2^n} \cdot \left(D' \cdot 0\right)^2 = 2^n \Rightarrow \\ \Rightarrow \operatorname{Var}\left(C\right) = 2^n - 1.$$

Is Knuth's algorithm always work? (Rare-events)

Consider the "hair brush" tree T and suppose that the costs of all vertices are zero except for v_{n+1}, which has a cost of unity.

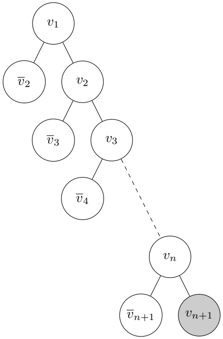


Figure: The hair brush tree.

The expectation and variance of the Knuth's estimator are

$$\mathbb{E}(C) = \frac{1}{2^n} \cdot 2^n \cdot 1 + \frac{2^n - 1}{2^n} \cdot D' \cdot 0 = 1,$$

and

$$\mathbb{E}\left(C^2\right) = \frac{1}{2^n} \cdot (2^n \cdot 1)^2 + \\ + \frac{2^n - 1}{2^n} \cdot \left(D' \cdot 0\right)^2 = 2^n \Rightarrow \\ \Rightarrow \mathrm{Var}(C) = 2^n - 1.$$

$$\mathrm{CV}^2 = \frac{\mathrm{Var}(\mathbf{C})}{\mathbb{E}(\mathbf{C})^2} = \frac{\mathbf{2^n - 1}}{\mathbf{1^2}}$$

Is Knuth's algorithm always work? (Rare-events)

Consider the "hair brush" tree T and suppose that the costs of all vertices are zero except for v_{n+1}, which has a cost of unity.

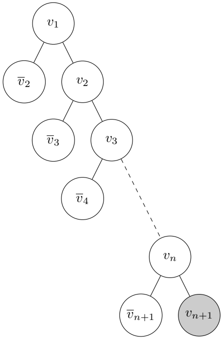


Figure: The hair brush tree.

The expectation and variance of the Knuth's estimator are

$$\mathbb{E}(C) = \mathbf{\frac{1}{2^n}} \cdot 2^n \cdot 1 + \frac{2^n - 1}{2^n} \cdot D' \cdot 0 = 1,$$

and

$$\mathbb{E}\left(C^2\right) = \frac{1}{2^n} \cdot (2^n \cdot 1)^2 + \\ + \frac{2^n - 1}{2^n} \cdot \left(D' \cdot 0\right)^2 = 2^n \Rightarrow \\ \Rightarrow \operatorname{Var}(C) = 2^n - 1.$$

$$\mathrm{CV}^{\mathbf{2}} = \frac{\operatorname{Var}(\mathbf{C})}{\mathbb{E}(\mathbf{C})^{\mathbf{2}}} = \frac{\mathbf{2^n - 1}}{\mathbf{1^2}}$$

What can we do?

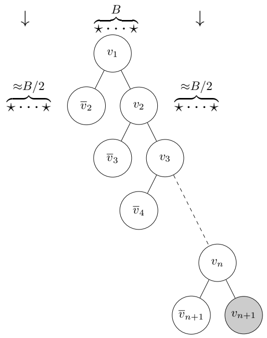
The problem is the large variance.

Variance reduction techniques.

- Common and antithetic random variables.
- Control variables.
- Conditional Monte Carlo.
- Stratified sampling.
- Importance Sampling.
- Multilevel Splitting.

To start with — Multilevel Splitting

Consider (again) the “hair brush” tree T.

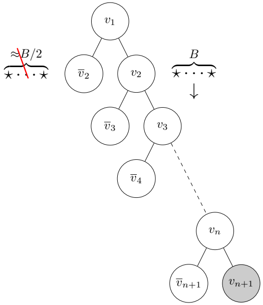


- Define some budget $B \geqslant 1$ of parallel random walks.
- Start from the root. The expected number of walks which reach the “good” vertex v_2 is $B/2$ — call them the “good” trajectories.
- Split the “good” trajectories such that there are B of them again and continue to the next tree level.
- Carefully choosing B (logarithmic in n), will allow us to reach the vertex of interest — v_{n+1} with reasonably high probability.

$$\mathbb{P}(\text{The process reaches the next level}) = 1 - 1/2^B.$$
$$\mathbb{P}(\text{The process reaches the } v_{n+1} \text{ vertex}) = (1 - 1/2^B)^n.$$
$$B = \log_2(n) \Rightarrow \mathbb{P}(\text{The process reaches the } v_{n+1} \text{ vertex}) \to e^{-1}, \text{ as } n \to \infty.$$

To start with — Multilevel Splitting

Consider (again) the "hair brush" tree T.

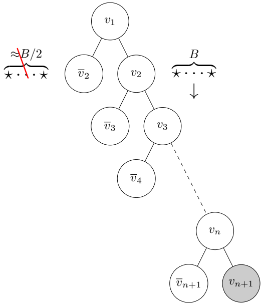


- Define some budget $B \geqslant 1$ of parallel random walks.
- Start from the root. The expected number of walks which reach the "good" vertex v_2 is $B/2$ — call them the "good" trajectories.
- Split the "good" trajectories such that there are B of them again and continue to the next tree level.
- Carefully choosing B (logarithmic in n), will allow us to reach the vertex of interest — v_{n+1} with reasonably high probability.

$$\mathbb{P}(\text{The process reaches the next level}) = 1 - 1/2^B.$$
$$\mathbb{P}(\text{The process reaches the } v_{n+1} \text{ vertex}) = (1 - 1/2^B)^n.$$
$$B = \log_2(n) \Rightarrow \mathbb{P}(\text{The process reaches the } v_{n+1} \text{ vertex}) \to e^{-1}, \text{ as } n \to \infty.$$

To start with — Multilevel Splitting

Consider (again) the "hair brush" tree T.



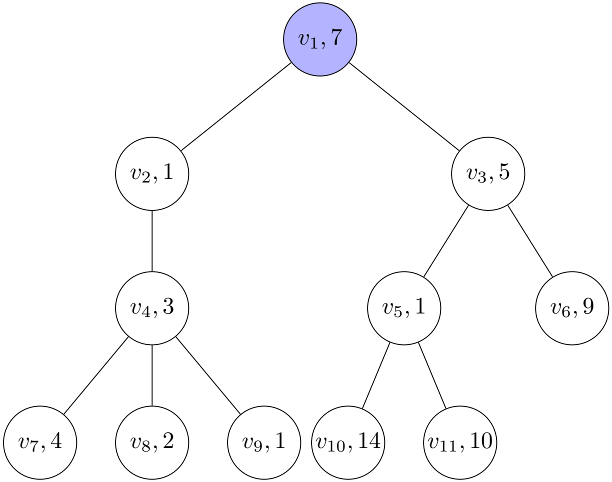
- Define some budget $B \geqslant 1$ of parallel random walks.
- Start from the root. The expected number of walks which reach the "good" vertex v_2 is $B/2$ — call them the "good" trajectories.
- Split the "good" trajectories such that there are B of them again and continue to the next tree level.
- Carefully choosing B (logarithmic in n), will allow us to reach the vertex of interest — v_{n+1} with reasonably high probability.

$$\mathbb{P}(\text{The process reaches the next level}) = 1 - 1/2^B.$$
$$\mathbb{P}(\text{The process reaches the } v_{n+1} \text{ vertex}) = (1 - 1/2^B)^n.$$
$$B = \log_2(n) \Rightarrow \mathbb{P}(\text{The process reaches the } v_{n+1} \text{ vertex}) \to e^{-1}, \text{ as } n \to \infty.$$

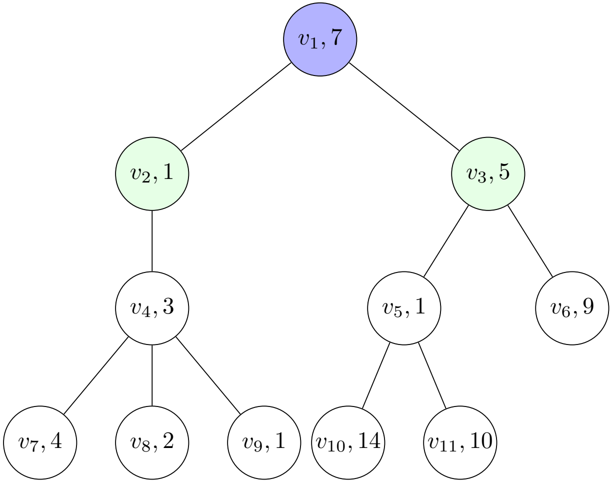
SE — the main idea

1. Define a budget $B \in \mathbb{N}$, and let B be the number of parallel random walks on the tree.
2. Using these B walks, run Knuth's Algorithm in parallel.
3. If some walks "die", split the remaining ones to continue with B walks as usual, (multilevel splitting).

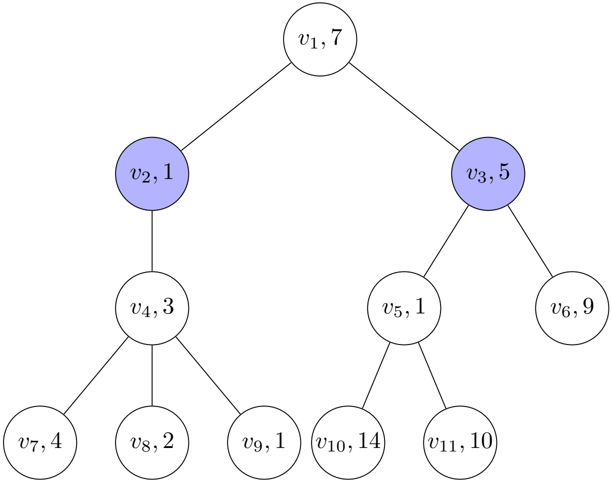
SE example with $B = 2$



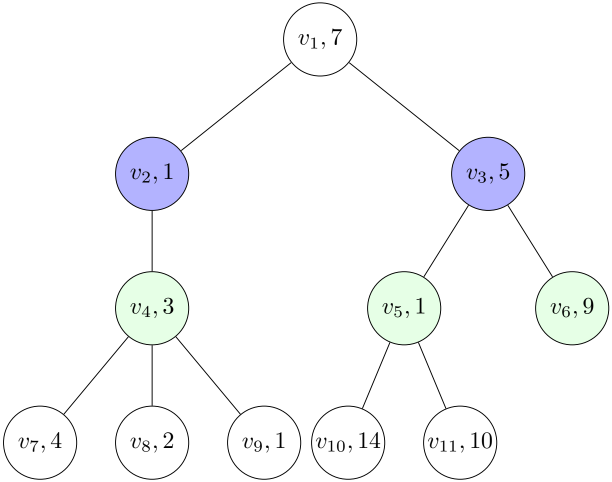
SE example with $B = 2$



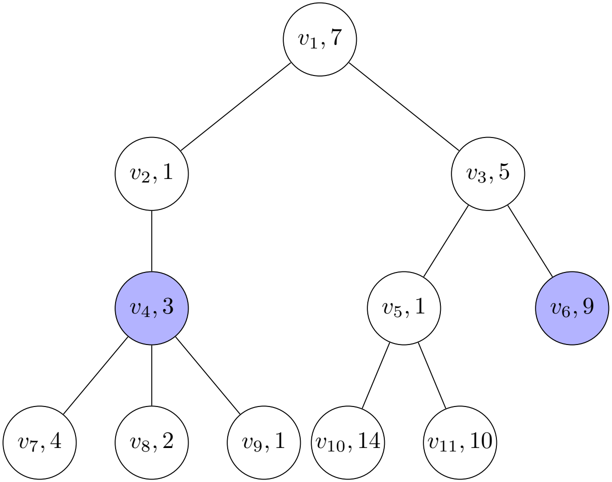
SE example with $B = 2$



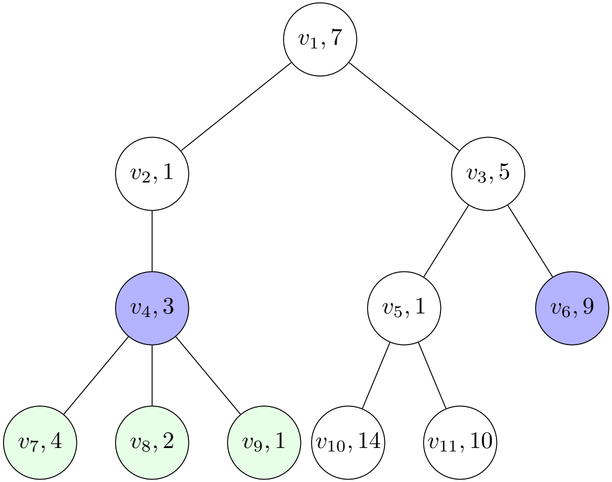
SE example with $B = 2$



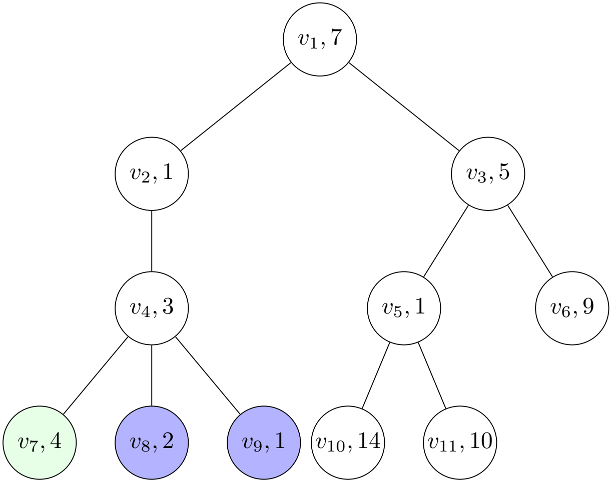
SE example with $B = 2$



SE example with $B = 2$



SE example with $B = 2$



A short summary

1. Many interesting problems from different research fields can be modeled by the tree counting problem.

2. Knuth's algorithm can fail badly.

3. Running several trajectories in parallel seems to introduce some added value.

4. Can we justify this rigorously?

The Analysis

Radidlav Vaisman and Dirk P. Kroese (2015) Stochastic Enumeration Method for Counting Trees. Methodology & Computing in Applied Probability.

Theorem (Basic Properties of SE)

Let T_v be tree rooted at v. Then,

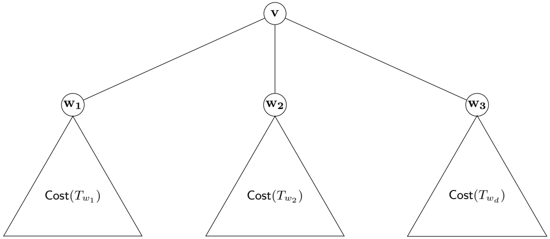
$$\mathbb{E}(C_{\mathrm{SE}}(T_v)) = \mathrm{Cost}(T_v),$$

and

$$\mathrm{Var}(C_{\mathrm{SE}}(T_{\mathbf{v}})) = \frac{\left(\frac{|S(\mathbf{v})|}{|\mathbf{v}|}\right)^2}{d} \sum_{1\leqslant j\leqslant d} \mathrm{Var}\left(C_{\mathrm{SE}}\left(T_{\mathbf{w}_j}\right)\right) + \frac{\left(\frac{|S(\mathbf{v})|}{|\mathbf{v}|}\right)^2}{d^2} \sum_{1\leqslant i<j\leqslant d} \left(\frac{\mathrm{Cost}\left(T_{\mathbf{w}_i}\right)}{|\mathbf{w}_i|} - \frac{\mathrm{Cost}\left(T_{\mathbf{w}_j}\right)}{|\mathbf{w}_j|}\right)^2,$$

where d is the degree of v, and $w_1, \cdots, w_d$ are the children of v.

An upper bound on SE's variance (1)



- We would like to calculate (or at least get an upper bound of), $\mathrm{Var}\left(C_{\mathrm{SE}}\left(T_{\mathbf{v}}\right)\right)$ for a general tree.
- This will also provide us with an upper bound on CV^2.

Analysis — upper bound on SE's variance (2)

Theorem

Suppose without loss of generality that there exists constant a such that

$$\frac{\mathrm{Cost}\left(T_{\mathbf{w}_1}\right)}{|\mathbf{w}_1|} \leqslant \frac{\mathrm{Cost}\left(T_{\mathbf{w}_2}\right)}{|\mathbf{w}_2|} \leqslant \cdots \leqslant \frac{\mathrm{Cost}\left(T_{\mathbf{w}_d}\right)}{|\mathbf{w}_d|} \leqslant a \frac{\mathrm{Cost}\left(T_{\mathbf{w}_1}\right)}{|\mathbf{w}_1|}.$$

Then, the variance of SE estimator satisfies

$$\mathrm{Var}\left(C_{\mathrm{SE}}\left(T_{\mathbf{v}}\right)\right) \leqslant \left(\beta^h - 1\right) \left[\frac{\mathrm{Cost}\left(T_{\mathbf{v}}\right)}{|\mathbf{v}|}\right]^2,$$

where $\beta = \left(\frac{a^2+2a+1}{4a}\right)$. *That is,* $\mathrm{CV}^2 \leqslant \beta^h - 1$.

- Is this good enough? — What if $\beta > 1$?
- Our numerical results nevertheless, showed great performance for various applications.

Random trees

Definition (Family of random trees)

Consider a probability vector $\mathbf{p} = (p_0, \dots, p_k)$ that corresponds to the probability of a vertex to have $0, \dots, k$ successors respectively. Define a family of random trees $\mathcal{F}_{\mathbf{p}}^h$ as all possible trees of hight at most h that are generated using $\mathbf{p}$ up to the level h.

- The family $\mathcal{F}_{\mathbf{p}}^h$ is fully characterized by the probability vector $\mathbf{p}$ and the parameter h.
- The tree generation corresponds to a branching process.

Objective

Let $T = (\mathcal{V}, \mathcal{E})$ be a random tree from $\mathcal{F}_{\mathbf{p}}^h$. By assigning the cost $c(v) = 1$ for all $v \in \mathcal{V}$, the cost of the tree — $\mathrm{Cost}(T)$ is equal to $|\mathcal{V}|$. Our objective is to analyse the behavior of Knuth's and SE's estimators under this setting.

Super–critical branching process

- Consider a random tree rooted at v_0 and let R_m be the total number of children (population size) at level (generation) m and denote by M_m the total progeny at generation m. Define

$$\mu = \mathbb{E}(R_1) = \sum_{0 \leqslant j \leqslant k} j p_j \quad \text{and} \quad \sigma^2 = \mathrm{Var}(R_1) = \left(\sum_{0 \leqslant j \leqslant k} j^2 p_j \right) - \mu^2.$$

- From [Pakes 1971], the total progeny satisfies:

$$\nu_m = \mathbb{E}(M_m) = \mathbb{E}\left(1 + \sum_{1 \leqslant j \leqslant m} R_t \right) = \frac{\mu^{m+1} - 1}{\mu - 1}.$$

Random trees — expected performance

Theorem (Knuth v.s SE)

For a random tree $T^{(h)}$ the following holds.

1. *Lower bound on Knuth's expected variance satisfies:*

$$\mathbb{E}\left(\mathrm{Var}\left(C\left(T^{(h)}\right) \;\middle|\; T^{(h)}\right)\right) \geqslant (\sigma^2+\mu^2-\mu)\,\frac{1-\left(\sigma^2+\mu^2\right)^h}{1-(\sigma^2+\mu^2)}.$$

2. *For*

$$B \geqslant \max\left\{\left\lceil \frac{hk^2\ln\left(2h(\sigma^2+\mu^2)\frac{\sigma^2\mu}{(\mu-1)^3}\right)}{2(\mu-1)^2}\right\rceil,\ \left\lceil\frac{h\sigma^2}{\mu^2}\right\rceil\right\},$$

the upper bound on SE's expected variance satisfies:

$$\mathbb{E}\left(\mathrm{Var}\left(C_{\mathrm{SE}}\left(T^{(h)}\right) \;\middle|\; T^{(h)}\right)\right) \leqslant B^2 he\mu^{2h}\left(\frac{\sigma^2\mu}{(\mu-1)^3}+1\right).$$

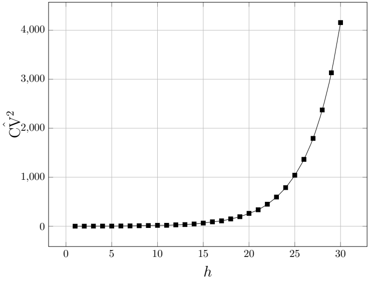
The expected variance reduction

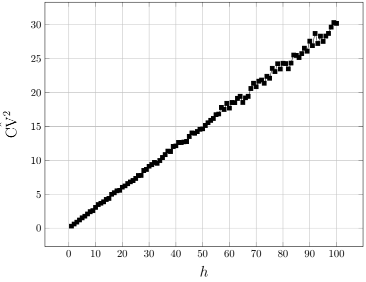
Corollary

The SE Algorithm introduces an expected variance reduction that is approximately equal to

$$\left(1 + \frac{\sigma^2}{\mu^2}\right)^h .$$

The variance reduction is exponential in h!





SE ia a Fully Polynomial Randomized Approximation Scheme

Theorem (Almost sure FPRAS)

Let $\mathcal{F}^h_{\mathbf{p}'}$ be a family of random trees such that for $T \in \mathcal{F}^h_{\mathbf{p}'}$

$$\lim_{h \to \infty} \mathbb{P}\left(\mathrm{Cost}(T) < \frac{1}{\mathrm{P}(h)} \nu_h\right) = 0,$$

where $\mathrm{P}(h) > 0$ is some polynomial function in h and $\nu_h = \frac{1-\mu^{h+1}}{1-\mu}$ is the expected number of nodes. In other words, for most instances, (almost surely), the actual number of nodes is not much smaller than the expectation. Then, under the above condition, and provided that

$$\mu > 1 + \delta \quad \text{for any } \delta > 0,$$

the SE algorithm is FPRAS for most of the instances in $T \in \mathcal{F}^h_{\mathbf{p}'}$, that is,

$$\mathrm{CV}^2 = \frac{\mathrm{Var}\left(C_{\mathrm{SE}}(T) \mid T\right)}{\left(\mathbb{E}\left(C_{\mathrm{SE}}(T) \mid T\right)\right)^2}$$

is bounded by a polynomial in h with high probability.

A short summary

1. Many interesting problems from different research fields can be modeled by the tree counting problem.
2. Knuth's algorithm can fail badly.
3. Running several trajectories in parallel is very beneficial. We obtained an FPRAS for random trees using SE.
4. Can I use the SE algorithm for practical problems?

SE in practice — Network Reliability, Sensitivity, and Cascades

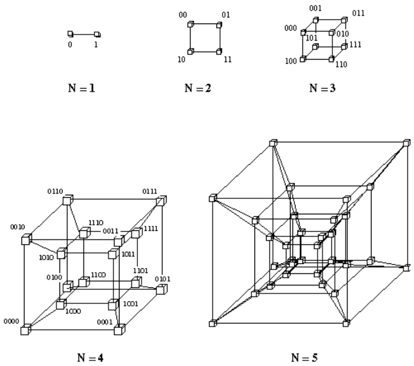
- Radidlav Vaisman, Dirk P. Kroese and Ilya B. Gertsbakh (2015) Improved Sampling Plans for Combinatorial Invariants of Coherent Systems, *IEEE transactions on reliability*.
- R. Solomone, R. Vaisman and D. P. Kroese (2016) Estimating the Number of Vertices in Convex Polytopes, Annual International Conference on Operations Research and Statistics, ORS.
- R. Shah and R. Vaisman (2016) New Sampling Plans for Estimating Residual Connectedness Reliability, Annual International Conference on Operations Research and Statistics, ORS.

SE in practice — Network Reliability, Sensitivity, and Cascades

- Radidlav Vaisman, Dirk P. Kroese and Ilya B. Gertsbakh (2015) Improved Sampling Plans for Combinatorial Invariants of Coherent Systems, *IEEE transactions on reliability*.
- R. Solomone, R. Vaisman and D. P. Kroese (2016) Estimating the Number of Vertices in Convex Polytopes, Annual International Conference on Operations Research and Statistics, ORS.
- R. Shah and R. Vaisman (2016) New Sampling Plans for Estimating Residual Connectedness Reliability, Annual International Conference on Operations Research and Statistics, ORS.

Estimating the Reliability Polynomial with SE (1)

- The hypercube graph H_n is a regular graph with 2^n vertices and $n2^{n-1}$ edges.
- In order to construct a hypercube graph, label every 2^n vertices with n-bit binary numbers and connect two vertices by an edge whenever the Hamming distance of their labels is 1.

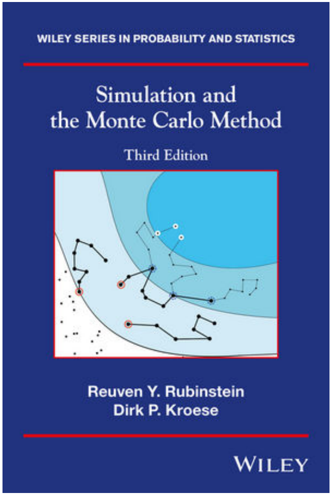


Estimating the Reliability Polynomial with SE (2)

- We consider H_5 with two terminals $K = \{0, 24\}$; that is (00000, 11000) in the binary representation.
- Using full enumeration procedure we found that for a reliable simulation, one should estimate values that are as small as $8.3195 \cdot 10^{-8}$.
- For this relatively small graph, the (usually used) Permutation Monte Carlo (PMC) algorithm needs huge sample size. Using $N = 10^9$ samples takes about 25 hours on my Core i5 laptop, The related error is about 60%. Why? The minimal value that must be estimated by PMC is $8.3195 \cdot 10^{-8}$ is rare event!
- The SE delivers very reliable estimates in 28 seconds with budget $B = 10$ and $N = 1000$. The related error is about 1%.

Further reading

Chapter 10.



WILEY SERIES IN PROBABILITY AND STATISTICS

Simulation and the Monte Carlo Method

Third Edition

Reuven Y. Rubinstein
Dirk P. Kroese

WILEY

What next?

- **(Hard)** — Finding more classes of trees that can be efficiently handled by SE, (that is, show proven performance guarantees like for the random tree case).
- **(In progress)** — Extending different Sequential Monte Carlo algorithms with SE mechanism (splitting).
- **(In progress)** — Adaptation of SE to (big) data analysis.
- **(In progress)** — Adaptation of SE to MDP/POMDP planning.

Part II

Advanced MC Methods for Network Reliability estimation

The Lomonosov's Turnip

The setup

- Given a network $G = G(V, E, K)$ where V is the node set, E is the edge set and $K \subseteq V$ is the terminal set.
- The edges states are binary; that is, edge e can be in the *up* or *down* state with probabilities p_e and $q_e = 1 - p_e$, respectively.

Note that we allow different edge failure probabilities.

- The network *UP* state is defined as the presence of connectivity of all terminal nodes.

We describe the PMC algorithm of Michael Lomonosov, also called the network evolution process.

The evolution process PMC

- The basic idea of PMC is to associate with each edge $e \in E$ an exponentially distributed random "birth time" $\tau(e)$ with parameter $\lambda(e)$, such that $\mathbb{P}(\tau(e) \leqslant \tau) = 1 - \mathrm{e}^{-\lambda(e)\tau} = p_e$ holds for all $e \in E$ and for an arbitrary chosen time value τ.
- Let us assume that all the edges are in the *down* state at time zero. Then, an edge e is born at time $\tau(e)$; that is, at the time $\tau(e)$ it enters the *up* state and stays there "forever".

Note that the probability that e will be "alive" at time τ is $\mathbb{P}(\tau(e) \leqslant \tau) = p_e$.

- The value of τ can be arbitrary, so for simplicity we put $\tau = 1$ and it follows that $\lambda(e) = -\ln q_e$.
- If we take a "snapshot" of the state of all edges at time instance $\tau = 1$, we will see the network in the state which is stochastically equivalent to the static picture in which edge e is *up* or *down* with probability p_e or q_e, respectively.

The evolution process PMC

- Suppose that $|E| = n$ and consider the ordering (permutation) of the edges $\boldsymbol{\pi} = (e_1, \ldots, e_n)$, according to their birth times sorted in increasing order.
- Since the birth times are exponentially distributed, (race of exponential r.v), it holds that

$$\mathbb{P}(\boldsymbol{\Pi} = \boldsymbol{\pi}) = \prod_{t=1}^{n} \frac{\lambda(e_t)}{\Lambda(E_{t-1})},$$

  where $E_t = E \setminus \{e_1, \ldots, e_t\}$ for $1 \leqslant t \leqslant n-1$, and $\Lambda(E_t) = \sum_{e \in E_t} \lambda(e)$.
- The first index $1 \leqslant \mathrm{a}(\boldsymbol{\pi}) \leqslant n$ of the edge permutation $\boldsymbol{\pi}$ for which the sub-graph of G defined by $G(V, (e_1, \ldots, e_{\mathrm{a}(\boldsymbol{\pi})}), K)$ is in the *UP* state, is called an anchor of $\boldsymbol{\pi}$. That is,

$$\mathrm{a}(\boldsymbol{\pi}) = \min \{t \ : \ G\,(V, (e_1, \ldots, e_t), K) \ \text{is } UP\}\,.$$

The evolution process

Let $\xi_1 + \cdots + \xi_t$ be the birth time of edge e_t in $\boldsymbol{\pi}$ for $1 \leqslant t \leqslant n$. Then, given the edge permutation $\boldsymbol{\Pi} = \boldsymbol{\pi}$, the probability that the network is in the *UP* state is given by

$$\mathbb{P}\left(\sum_{t=1}^{\mathrm{a}(\boldsymbol{\pi})} \xi_t \leqslant 1 \,\middle|\, \boldsymbol{\Pi} = \boldsymbol{\pi}\right) = \mathrm{Conv}_{1 \leqslant t \leqslant \mathrm{a}(\boldsymbol{\pi})} \left\{1 - \mathrm{e}^{-\Lambda(E_t)}\right\},$$

where Conv stands for exponential convolution. The network *DOWN* and *UP* probabilities denoted by $\bar{r}$ and r, respectively, can be expressed as

$$\bar{r} = \sum_{\boldsymbol{\pi}} \mathbb{P}(\boldsymbol{\Pi} = \boldsymbol{\pi}) \cdot \mathbb{P}\left(\sum_{t=1}^{\mathrm{a}(\boldsymbol{\pi})} \xi_t > 1 \,\middle|\, \boldsymbol{\Pi} = \boldsymbol{\pi}\right),$$

and

$$r = \sum_{\boldsymbol{\pi}} \mathbb{P}(\boldsymbol{\Pi} = \boldsymbol{\pi}) \cdot \mathbb{P}\left(\sum_{t=1}^{\mathrm{a}(\boldsymbol{\pi})} \xi_t \leqslant 1 \,\middle|\, \boldsymbol{\Pi} = \boldsymbol{\pi}\right),$$

respectively, where the summation is over all permutations $\boldsymbol{\pi}$.

The evolution process

Since the network unreliability and reliability in

$$\bar{r} = \sum_{\boldsymbol{\pi}} \mathbb{P}(\boldsymbol{\Pi} = \boldsymbol{\pi}) \cdot \mathbb{P}\left(\sum_{t=1}^{\mathsf{a}(\boldsymbol{\pi})} \xi_t > 1 \;\middle|\; \boldsymbol{\Pi} = \boldsymbol{\pi} \right),$$

and

$$r = \sum_{\boldsymbol{\pi}} \mathbb{P}(\boldsymbol{\Pi} = \boldsymbol{\pi}) \cdot \mathbb{P}\left(\sum_{t=1}^{\mathsf{a}(\boldsymbol{\pi})} \xi_t \leqslant 1 \;\middle|\; \boldsymbol{\Pi} = \boldsymbol{\pi} \right),$$

is expressed as an expectation, it can be estimated without bias as the sample average of conditional probabilities, $\mathbb{P}(\xi_1 + \xi_2 + \cdots + \xi_{\mathsf{a}(\boldsymbol{\Pi})} > 1 \mid \boldsymbol{\Pi})$ over an independent sample of trajectories $\{\boldsymbol{\Pi}^{(1)}, \boldsymbol{\Pi}^{(2)}, \ldots, \boldsymbol{\Pi}^{(N)}\}$.

PMC Algorithm For Unreliability Estimation

Given a network $G = G(V, E, K)$, edge failure probabilities $(q_e,\ e \in E)$, and sample size N, execute the following steps.

1. **(Initialization)** Set $S \leftarrow 0$. For each edge $e \in E$, set $\lambda(e) \leftarrow -\ln(q_e)$ and $k \leftarrow 0$.
2. **(Permutation Generation)** Set $k \leftarrow k+1$ and sample $\boldsymbol{\Pi}^{(k)} = \left(e_1^{(k)}, \ldots, e_n^{(k)}\right)$.
3. **(Find the Anchor)** Calculate
$$\mathsf{a}\left(\boldsymbol{\Pi}^{(k)}\right) = \min\left\{t \ :\ G\left(V, \left(e_1^{(k)}, \ldots, e_t^{(k)}\right), K\right) \text{ is UP}\right\}.$$
4. **(Calculation of Convolution)** Set:
$$R^{(k)} \leftarrow 1 - \mathrm{Conv}_{1 \leqslant t \leqslant \mathsf{a}(\boldsymbol{\Pi}^{(k)})}\left\{1 - \mathrm{e}^{-\Lambda\left(E_t^{(k)}\right)}\right\},$$
where $E_t^{(k)} = E \setminus \left\{e_1^{(k)}, \ldots, e_t^{(k)}\right\}$ for $1 \leqslant t \leqslant \boldsymbol{\Pi}^{(k)}$, and $\Lambda\left(E_t^{(k)}\right) = \sum_{e \in E_t^{(k)}} \lambda(e)$, and set $S \leftarrow S + R^{(k)}$.
5. **(Stopping Condition)** If $k = N$, return S/N as unbiased estimator of $\bar{r}$; otherwise, go to Step 2.

PMC Algorithm For Unreliability Estimation — problems

- The main issue with the PMC algorithm, is its non-uniform trajectory generation with respect to their length.
- Namely, shorter trajectories, which have a small anchor, have a bigger chance to be generated during the evolution process.
- However, long trajectories will generally have higher weights in the estimator. This issue causes a considerable increase in PMC estimator variance.

PMC Algorithm For Unreliability Estimation — problems

- Lomonosov tried to resolve this issue by equipping the PMC Algorithm with a so-called closure (merging) operation.
- The closure of a subset $E' \subseteq E$ consists of E' and all edges of G whose vertices lie in the same component of the spanning subgraph $G(V, E')$.
- The closure operation is essentially an elimination of edges that do not change the already born connected component during the evolution process.
- With this addition, the PMC algorithm has a higher chance of generating long trajectories. The corresponding estimator was shown to be unbiased and its relative error is uniformly bounded with respect to the $\lambda(e)$ values.

Lomonosov's Turnip (LT)

- To implement the merging process, all that needs to be done after each birth of an edge, is to look for those edges whose nodes belong to the already existing component.

- These edges are joined to this component and excluded from further considerations as irrelevant.

- This combination of merging and the evolution process causes the reliability estimator to become less variable and is called the LT algorithm.

Lomonosov's Turnip problems

- Consider a simple network $\mathfrak{S}(n)$ with $n+2$ nodes and $2n+1$ edges presented below.
- The terminal set consists of two vertices, u and v. For this particular network topology, the closure operation has no effect, since during the edge birth process no edge can be merged and thus the LT algorithm turns into the regular PMC.

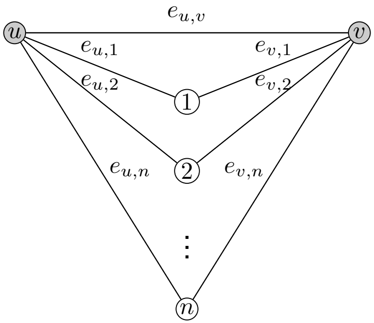


Figure: A simple network $\mathfrak{S}(n)$ with $n+2$ nodes, $2n+1$ edges, and $K = \{u, v\}$.

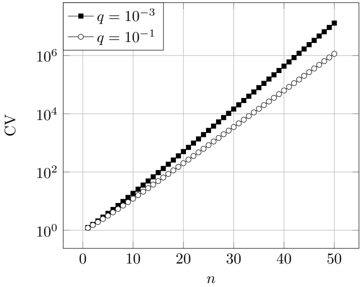
Lomonosov's Turnip problems

- Suppose, for example, that each edge fails with same probability $q = 1 - p$. Then, the $u - v$ network unreliability $\bar{r}$ is given by $(1-p)(1-p^2)^n$.
- Consider the distribution of the anchor for this particular network structure. Let $T = \mathrm{a}(\mathbf{\Pi})$ be the random variable that stands for the anchor.
- The LT algorithm returns $T = 1$ if the edge between u and v is the first one that enters the *up* state; that is, if $e_{u,v}$ is born first.
- The probability that the birth process stops after the birth of the t-th edge is (need some working to show that)

$$\mathbb{P}(T = t) = \frac{t}{2n+2-t} \prod_{j=1}^{t-1} \left(1 - \frac{j}{2n+2-j}\right), \quad t = 1, \ldots, n+1.$$

Lomonosov's Turnip problems

The exact value of the reliability and the analytical expression for $\mathbb{P}(T = t)$ allow to get an important insight about the coefficient of variation of the LT estimator.



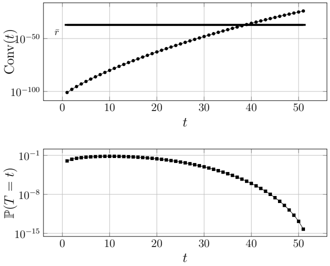


Figure: Left panel: logarithmically scaled CV of LT as a function of n for $\mathfrak{S}(n)$ networks. Right panel: $\mathrm{Conv}(t)$ and $\mathbb{P}(T = t)$ as a function of t for $\mathfrak{S}(50)$ and $p = 0.9$. The true unreliability of 8.66×10^{-38} is given by the horizontal line in the upper plot.

Interpretation

- This phenomenon happens as a result of the rare-event involvement.
- The right panel of the Figure shows the convolution $\mathrm{Conv}(t)$ and the probability $\mathbb{P}(T = t)$ as a function of anchor $1 \leqslant t \leqslant 51$.
- We can see that the long trajectories contribute the most mass to the estimator; but these long trajectories appear with very small probabilities.
- For example, we found that the average trajectory length is about 11.69. However, the trajectories that contribute most to the estimator are of length greater than 40.
- These trajectories are generated with a probability of less than 10^{-6}. This issue can be clearly observed in the upper plot of the right panel of the Figure by noting that the intersection of the horizontal line (which represents the true unreliability) and the convolution curve, occurs near $t = 40$.
- Long trajectories are generated with very small probabilities, as can be verified from the bottom plot of the right panel and thus the resulting estimator tends to underestimate the true value of interest.

To overcome the problem presented in the above example, we propose to combine the LT algorithm with the splitting method, which was proved to be very useful when working in a rare-event setting.

Consider a random variable (vector) $\mathbf{X}$ taking values in a set $\mathcal{X}$. A general objective of Monte Carlo simulation is to calculate $\ell = \mathbb{E}_f\left(H\left(\mathbf{X}\right)\right)$, where $H : \mathcal{X} \to \mathbb{R}$ is a real-valued function. The Crude Monte Carlo (CMC) estimator of ℓ is given by

$$\widehat{\ell} = \frac{1}{N}\sum_{k=1}^{N} H\left(\mathbf{X}^{(k)}\right),$$

where $\mathbf{X}^{(k)}$ for $k = 1, \ldots, N$, are independent copies of a random variable $\mathbf{X}$ generated from $f(\mathbf{x})$.

The Sequential MC (SMC) framework

- Suppose that the vector $\mathbf{X} \in \mathcal{X}$ is decomposable and that it can be of different length $T \in \{1, \dots, n\}$, where T is a stopping time of $\mathbf{X}$'s generation process.
- Thus, $\mathbf{X}$ can be written as $\mathbf{X} = (X_1, X_2, \dots, X_T)$, where for each $t = 1, \dots, T$, X_t can be multidimensional.
- We assume that $\mathbf{X}$ can be constructed sequentially such that its probability density function (PDF) $f(\mathbf{x})$ constitutes a product of conditional PDFs:

$$f(\mathbf{x}) = f_1(x_1) f_2(x_2 \mid x_1) \cdots f_t(x_t \mid x_1, \dots, x_{t-1}), |\mathbf{x}| = t, \ t = 1, \dots, n,$$

  where $|\mathbf{x}|$ is the length of $\mathbf{x}$.

The Sequential MC (SMC) framework

- This setting frequently occurs in practice. For example, consider a coin that is tossed repeatedly until the first "success" (1) appears or until n tosses have been made. The sample space is equal to

$$\mathcal{X} = \{(1),\ (0,1),\ \ (0,0,1), \ldots, (\underbrace{0,\ldots,0,}_{n-1 \text{ times}} 1),\ \ (\underbrace{0,\ldots,0}_{n \text{ times}})\}.$$

- That is, the samples have different lengths: $t = 1, 2, 3, \ldots, n$. Let $\mathcal{X}_t = \{\mathbf{x} \in \mathcal{X} : |\mathbf{x}| = t\}$ be the set of all samples of length $t = 1, 2, \ldots, n$.
- Then, the sets $\mathcal{X}_1, \ldots, \mathcal{X}_n$ define a partition of $\mathcal{X}$; that is

$$\mathcal{X} = \bigcup_{t=1}^{n} \mathcal{X}_t, \quad \mathcal{X}_{t_1} \cap \mathcal{X}_{t_2} = \emptyset \quad \text{for } 1 \leqslant t_1 < t_2 \leqslant n.$$

The Sequential MC (SMC) framework

- Since we are working under the SMC framework, the generation of
$$\mathbf{X} = (X_1, \ldots, X_T) \in \mathcal{X}_T,$$
is sequential in the following sense.
- We start from the "empty" $\mathbf{X} = ()$.
- Then X_1 is sampled from $f_1(x_1)$ and at each step $t \geqslant 2$, we sample X_t from $f_t(x_t \mid x_1, \ldots, x_{t-1})$ until the stopping time T that is determined from the generated X_t's.
- This procedure terminates at time $1 \leqslant T \leqslant n$ if $\mathbf{X} \in \mathcal{X}_T$.

Crude Sequential Monte Carlo (CSMC)

Given the density $f(\mathbf{x}) = f_1(x_1)f_2(x_2 \mid x_1)\cdots f_t(x_t \mid x_1, \ldots, x_{t-1})$, $\mathbf{x} \in \mathcal{X}_t$, $t = 1, \ldots, n$, and $H : \mathcal{X} \to \mathbb{R}$, output Z — an unbiased estimator of $\mathbb{E}_f\,(H\,(\mathbf{X}))$.

1. **(Initialization)** Set $t \leftarrow 0$ and $\mathbf{X} \leftarrow ()$.
2. **(Simulate and Update)** Set $t \leftarrow t + 1$, sample $X_t \sim f_t(x_t \mid X_1, \ldots, X_{t-1})$, and set $\mathbf{X} \leftarrow (X_1, \ldots, X_{t-1}, X_t)$.
3. **(Stopping Condition)** If $T = t$ (the stopping condition which can be determined from $\mathbf{X} = (X_1, \ldots, X_t)$), output $Z \leftarrow H(\mathbf{X})$; otherwise, go to Step 2.

The PMC algorithm is CSMC!

- To see that the PMC and the LT Algorithms are aligned with the SMC framework described above, let $\mathbf{X} = (\Pi_1, \ldots, \Pi_T)$, with $T = \mathrm{a}(\Pi)$, and

$$H(\mathbf{x}) = 1 - \mathrm{Conv}_{1 \leqslant t \leqslant T} \left\{ 1 - \mathrm{e}^{-\Lambda(E_i)} \right\}.$$

- Moreover, $f(\mathbf{x})$ is distributed according to (2), which is of the product form

$$f(\mathbf{x}) = \prod_{j=1}^{t} f_j, \quad 1 \leqslant t \leqslant n,$$

where f_j is defined by

$$f_j(\Pi_j = e_j \mid \Pi_1 = e_1, \ldots, \Pi_{j-1} = e_{j-1}) = \frac{\lambda(e_j)}{\Lambda(E_{j-1})}.$$

SMC

Keep in mind the PMC algorithm and the short trajectories phenomena.

For the forthcoming discussion, it will be convenient to define an event

$$\{\text{the SMC generation process did not stop at steps } 1, \ldots, t\} = \{T > t\}.$$

The $\mathbf{X}$ generation stochastic process can be visualized using the Figure on the next slide.

A random walk starts from the root of the tree $\{T > 0\}$ and ends at one of tree leaves $\{T = 1\}, \ldots, \{T = n\}$.

The SMC process

$$\begin{tikzcd}
 & \boxed{T > 0} \arrow[dl, "\mathbb{P}(T = 1 \mid T > 0)"'] \arrow[dr, "\mathbb{P}(T > 1 \mid T > 0)"] & & & & \\
\boxed{T = 1} & & \boxed{T > 1} \arrow[dl, "\mathbb{P}(T = 2 \mid T > 1)"'] \arrow[dr, "\mathbb{P}(T > 2 \mid T > 1)"] & & & \\
 & \boxed{T = 2} & & \boxed{T > 2} \arrow[dl, "\mathbb{P}(T = 3 \mid T > 2)"'] \arrow[ddr, dashed] & & \\
 & & \boxed{T = 3} & & & \\
 & & & & \boxed{T > n - 2} \arrow[dl, "\mathbb{P}(T = n - 1 \mid T > n - 2)"'] \arrow[dr, "\mathbb{P}(T > n - 1 \mid T > n - 2)"] & \\
 & & & \boxed{T = n - 1} & & \boxed{T > n - 1} \arrow[d, "\mathbb{P}(T = n \mid T > n - 1)"'] \\
 & & & & & \boxed{T = n}
\end{tikzcd}$$

SMC

- The splitting mechanism allows the process to reach $\{T = n\}$ with reasonably high probability, while using a relatively small budget B which can be logarithmic in the tree height.
- Note that the probability that this splitting process reaches the level t given that it reached level $t-1$ is $\mathbb{P}(T = t \mid T > t-1) = 1 - 1/2^B$, and as a consequence, the probability to reach $\{T = n\}$ is equal to

$$\mathbb{P}\left(\text{The splitting reaches the } \{T = n\} \text{ leaf}\right) = \left(1 - 1/2^B\right)^{n-1}.$$

- To conclude, the choice of $B = \lceil \log_2(n-1) \rceil$ results in

$$\mathbb{P}\left(\text{The process reaches } \{T = n\}\right) \geqslant (1 - 1/2^{\log_2(n-1)})^{n-1} \to e^{-1}$$

as $n \to \infty$.

Splitting Sequential Monte Carlo (SSMC) (1)

Given the density $f(\mathbf{x}) = f_1(x_1) f_2(x_2 \mid x_1) \cdots f_t(x_t \mid x_1, \ldots, x_{t-1})$, for $\mathbf{x} \in \mathcal{X}_t$, $1 \leqslant t \leqslant n$, $H : \mathcal{X} \to \mathbb{R}$, and a budget $B \in \mathbb{N} \setminus \{0\}$, output C — an unbiased estimator of $\mathbb{E}_f\left(H\left(\mathbf{X}\right)\right)$.

1. **(Initialization)** Set $t \leftarrow 0$, $P_t \leftarrow 1$ — an estimator of $\mathbb{P}\left(T > t\right)$, $C \leftarrow 0$, and define

$$\mathcal{W}^{(t)} = \left\{\mathbf{X}_1^{(t)}, \ldots, \mathbf{X}_B^{(t)}\right\},$$

where $\mathbf{X}_j^{(t)} \leftarrow ()$ for $j = 1, \ldots, B$, which we call the "working" set, because it contains unfinished trajectories.

Splitting Sequential Monte Carlo (SSMC) (2)

2. **(Simulate and Update)** Set $t \to t+1$. For each $\mathbf{X} = (X_1, \ldots, X_{t-1}) \in \mathcal{W}^{(t-1)}$, sample

$$X_t \sim f_t(x_t \mid X_1, \ldots, X_{t-1}),$$

and update: $\mathbf{X} \leftarrow (X_1, \ldots, X_t)$. Update the "finished" and "working" sets:

$$\mathcal{F}^{(t)} \leftarrow \left\{ \mathbf{X} \in \mathcal{W}^{(t-1)} \ : \ \mathbf{X} \in \mathcal{X}_t \right\}, \quad B_t \leftarrow \left| \mathcal{F}^{(t)} \right|,$$

$$\mathcal{W}^{(t)} \leftarrow \left\{ \mathbf{X} \in \mathcal{W}^{(t-1)} \ : \ \mathbf{X} \notin \mathcal{X}_t \right\}, \quad B'_t \leftarrow \left| \mathcal{W}^{(t)} \right|.$$

If $B_t = 0$, go to Step 2; otherwise, set

$$C_t \leftarrow P_{t-1} \frac{1}{B} \sum_{\mathbf{X} \in \mathcal{F}^{(t)}} H(\mathbf{X}), \quad C \leftarrow C + C_t, \quad P_t \leftarrow P_{t-1} \frac{B'_t}{B}.$$

Splitting Sequential Monte Carlo (SSMC) (3)

3. **(Stopping Condition)** If $B'_t = 0$, output C as an estimator of $\mathbb{E}_f(H(\mathbf{X}))$.

4. **(Splitting)** Insert K_j copies of each $\mathbf{X}_j \in \mathcal{W}^{(t)}$ into $\mathcal{W}^{(t)}$, where K_j satisfies

$$K_j = \lfloor B/B'_t \rfloor + L_j,$$

and $L_j \sim \text{Ber}(0.5)$ conditional on $\sum_{s=1}^{B'_t} L_s = B \bmod B'_t$. Go to Step 2.

Analysis

Theorem (Unbiased estimator)

The SSMC Algorithm outputs an unbiased estimator; that is, it holds that

$$\mathbb{E}(C) = \mathbb{E}_f \left(H(\mathbf{X}) \right).$$

- Although it is generally hard to analyze the efficiency of SSMC for a given problem in terms of RE, we provide performance guaranties under some simplified assumptions.
- However, it is important to note that the unbiasedness holds for general SMC procedures which can be presented in the form of CSMC.

Analysis

Theorem (Efficiency of SSMC Algorithm)

Suppose that the following holds for all $t = 1, \ldots, n$.

1. *For* $t = 1, \ldots, n$, $f_t(x_t \mid x_1, \ldots, x_{t-1}) = p_t$ *for all* $\mathbf{x} = (x_1, \ldots, x_{t-1}) \in \mathcal{X}_{t-1}$, *and* $p_t = \mathcal{O}(1/\mathcal{P}_n)$, *where* $\mathcal{P}_n$ *is a polynomial in* n.
2. $H(\mathbf{x}) = H_t$ *(constant) for all* $\mathbf{x} \in \mathcal{X}_t$, $t = 1, \ldots, n$.

Then, under above assumptions, the SSMC Algorithm is efficient; that is, it holds that $\mathrm{CV} = \sqrt{\mathrm{Var}\,(C)}/\mathbb{E}\,(C)$ *is upper-bounded by a polynomial in* n.

Corollary (Efficiency of SSMC for $\mathfrak{S}(n)$ networks)

The PMC Algorithm combined with SSMC, is an FPRAS for networks $\mathfrak{S}(n)$, $n > 0$.

Experiments — the graph $\mathfrak{S}(n)$

For the ST algorithm, we set $B = 1000$ and $N = 100$. Consequentially, we use $N = 10^5$ sample size for the LT algorithm. Table 3 summarizes the average performance of LT and ST for the $\mathfrak{S}(50)$ network using the above parameters. The bad performance of LT is not very surprising, since we know that for $\mathfrak{S}(50)$, the CV is of order 10^6.

Algorithm	$\overline{R}$	$\widehat{\mathrm{RE}}$	REE
LT	1.93×10^{-41}	76.5%	99.7%
ST	8.67×10^{-38}	2.48%	2.41%

Table: The performance obtained for the $\mathfrak{S}(50)$ network with $p = 0.9$ using the LT and the ST algorithms. The true unreliability is 8.66×10^{-38}.

Experiments — the dodecahedron graph

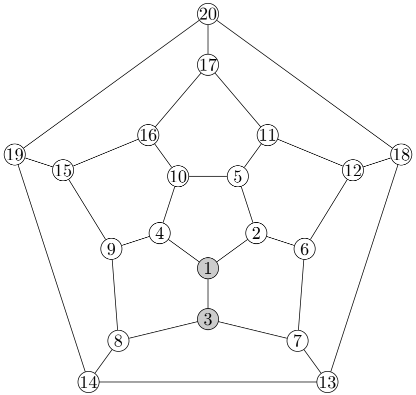


Figure: The dodecahedron graph with 20 vertices, 30 edges, and $K = \{1, 3\}$.

Experiments — the dodecahedron graph

	LT		ST	
q	$\overline{R}$	$\widehat{\mathrm{RE}}$	$\overline{R}$	$\widehat{\mathrm{RE}}$
10^{-15}	6.35×10^{-31}	58.1%	3.24×10^{-30}	5.04%
10^{-14}	8.80×10^{-29}	56.9%	3.25×10^{-28}	4.77%
10^{-13}	1.49×10^{-26}	55.3%	3.25×10^{-26}	4.18%
10^{-12}	6.85×10^{-25}	50.7%	3.24×10^{-24}	3.78%
10^{-11}	1.39×10^{-22}	50.7%	3.23×10^{-22}	3.45%
10^{-10}	1.01×10^{-20}	49.9%	3.22×10^{-20}	3.00%
10^{-9}	4.94×10^{-18}	48.2%	3.24×10^{-18}	2.81%
10^{-8}	2.14×10^{-16}	47.8%	3.23×10^{-16}	2.50%
10^{-7}	2.50×10^{-14}	43.8%	3.23×10^{-14}	2.22%
10^{-6}	4.30×10^{-12}	42.5%	3.24×10^{-12}	1.96%
10^{-5}	3.46×10^{-10}	32.8%	3.24×10^{-10}	1.66%
10^{-4}	3.18×10^{-8}	17.9%	3.24×10^{-8}	1.36%
10^{-3}	3.24×10^{-6}	6.68%	3.24×10^{-6}	1.23%
10^{-2}	3.20×10^{-4}	1.85%	3.20×10^{-4}	0.75%
10^{-1}	2.82×10^{-2}	0.43%	2.82×10^{-2}	0.42%

Table: A summary of average performance obtained for the dodecahedron network using the LT and ST algorithms.