Efficient Sequential Importance Sampling for Counting Vertex Covers via Stochastic Relaxation Technique

Radislav (Slava) Vaisman

(joint work with Zdravko Botev (UNSW) and
Ad Ridder (Vrije Universiteit))

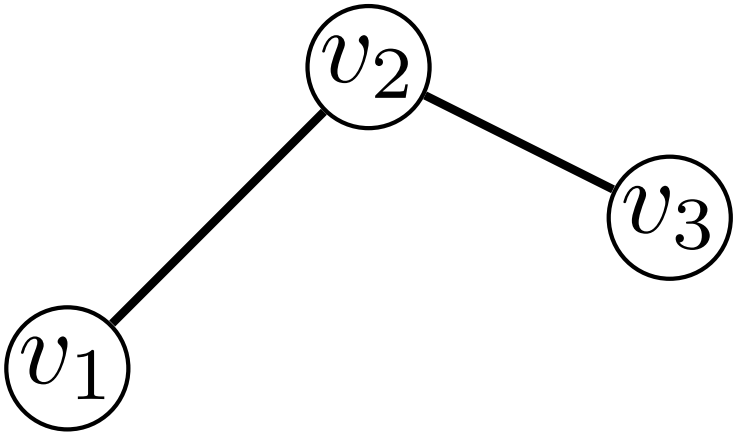
School of Mathematics and Physics
The University of Queensland Australia
https://www.smp.uq.edu.au/people/RadislavVaisman
r.vaisman@uq.edu.au

November, 2015

Overview

Vertex Cover

A vertex cover of a graph is a set of vertices such that each edge of the graph is incident to at least one vertex of the set.



$$\mathrm{VC} = \{\{v_2\}, \{v_1, v_3\}, \ldots, \{v_1, v_2, v_3\}\}.$$

Hardness of the vertex cover problem

- Finding minimum vertex cover is NP-hard.

Hardness of the vertex cover problem

- Finding minimum vertex cover is NP-hard.
- Counting all vertex covers is #P.

Hardness of the vertex cover problem

- Finding minimum vertex cover is NP-hard.
- Counting all vertex covers is #P.
- Counting the number of vertex covers remains hard even when restricted to planar bipartite graphs of bounded degree or regular graphs of constant degree.

Hardness of the vertex cover problem

- Finding minimum vertex cover is NP-hard.
- Counting all vertex covers is #P.
- Counting the number of vertex covers remains hard even when restricted to planar bipartite graphs of bounded degree or regular graphs of constant degree.

- Proposed solution — Sequential Importance Sampling (SIS).

  R. Vaisman, Z. I. Botev and A. Ridder, Sequential Monte Carlo for Counting Vertex Covers in General Graphs, Statistics and Computing, 2014.

A basic SIS Algorithm

Given a graph $G = (V, E)$, $|V| = n$, execute the following steps.

A basic SIS Algorithm

Given a graph $G = (V, E)$, $|V| = n$, execute the following steps.

1. Define a graph vertex ordering $(v_1, \ldots, v_n)$, set $\mathbf{C} = \emptyset$, and set $X = 1$.

A basic SIS Algorithm

Given a graph $G = (V, E)$, $|V| = n$, execute the following steps.

1. Define a graph vertex ordering $(v_1, \ldots, v_n)$, set $\mathbf{C} = \emptyset$, and set $X = 1$.
2. For each $i = 1, \ldots, n$, flip a $\tilde{p}_i - \tilde{q}_i$ coin.
   - If heads: take v_i to the cover ($\mathbf{C} = \mathbf{C} \cup \{v_i\}$), and set $X = \tilde{p}_i^{-1} X$,
   - else, set $X = \tilde{q}_i^{-1} X$,

A basic SIS Algorithm

Given a graph $G = (V, E)$, $|V| = n$, execute the following steps.

1. Define a graph vertex ordering $(v_1, \ldots, v_n)$, set $\mathbf{C} = \emptyset$, and set $X = 1$.
2. For each $i = 1, \ldots, n$, flip a $\tilde{p}_i - \tilde{q}_i$ coin.
   - If heads: take v_i to the cover ($\mathbf{C} = \mathbf{C} \cup \{v_i\}$), and set $X = \tilde{p}_i^{-1} X$,
   - else, set $X = \tilde{q}_i^{-1} X$,
3. If $\mathbf{C}$ does not form a valid vertex cover, set $X = 0$. Return X as an estimator of the number of vertex covers in the $G = (V, E)$ graph.

A basic SIS Algorithm

Given a graph $G = (V, E)$, $|V| = n$, execute the following steps.

1. Define a graph vertex ordering $(v_1, \ldots, v_n)$, set $\mathbf{C} = \emptyset$, and set $X = 1$.
2. For each $i = 1, \ldots, n$, flip a $\tilde{p}_i - \tilde{q}_i$ coin.
   - If heads: take v_i to the cover ($\mathbf{C} = \mathbf{C} \cup \{v_i\}$), and set $X = \tilde{p}_i^{-1} X$,
   - else, set $X = \tilde{q}_i^{-1} X$,
3. If $\mathbf{C}$ does not form a valid vertex cover, set $X = 0$. Return X as an estimator of the number of vertex covers in the $G = (V, E)$ graph.

Theorem (Estimator unbiasedness)

For **any** $\tilde{p}_i$ *and* $\tilde{q}_i = 1 - \tilde{p}_i$ $(i = 1, \ldots, n)$, *it holds that:*

$$\mathbb{E}(X) = \mathrm{VC}(G) = \#\ \textit{of vertex covers in}\ G.$$

A basic SIS Algorithm

Given a graph $G = (V, E)$, $|V| = n$, execute the following steps.

1. Define a graph vertex ordering $(v_1, \ldots, v_n)$, set $\mathbf{C} = \emptyset$, and set $X = 1$.
2. For each $i = 1, \ldots, n$, flip a $\tilde{p}_i - \tilde{q}_i$ coin.
   - If heads: take v_i to the cover ($\mathbf{C} = \mathbf{C} \cup \{v_i\}$), and set $X = \tilde{p}_i^{-1} X$,
   - else, set $X = \tilde{q}_i^{-1} X$,
3. If $\mathbf{C}$ does not form a valid vertex cover, set $X = 0$. Return X as an estimator of the number of vertex covers in the $G = (V, E)$ graph.

Theorem (Estimator unbiasedness)

For **any** $\tilde{p}_i$ *and* $\tilde{q}_i = 1 - \tilde{p}_i$ $(i = 1, \ldots, n)$, *it holds that:*

$$\mathbb{E}(X) = \mathrm{VC}(G) = \#\ \textit{of vertex covers in } G.$$

What is the best way to choose the probability $\tilde{p}_i$ for $i = 1, \ldots, n$?

The Importance Sampling

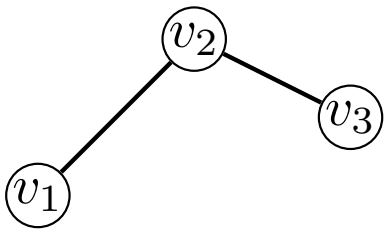
Choose:

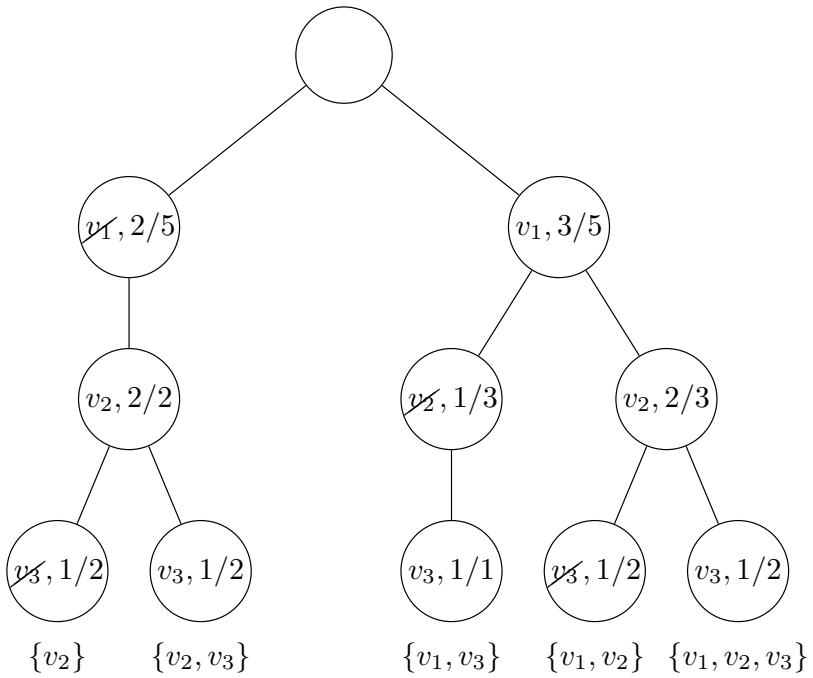
$$\tilde{p}_i^* = \frac{|\{\text{valid covers which contain } \mathbf{C} \cup \{v_i\}\}|}{|\{\text{valid covers which contain } \mathbf{C}\}|} \quad \forall i = 1, \ldots, n.$$

The Importance Sampling

Choose:

$$\tilde{p}_i^* = \frac{|\{\text{valid covers which contain } \mathbf{C} \cup \{v_i\}\}|}{|\{\text{valid covers which contain } \mathbf{C}\}|} \quad \forall i = 1, \ldots, n.$$

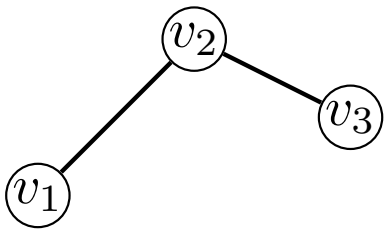


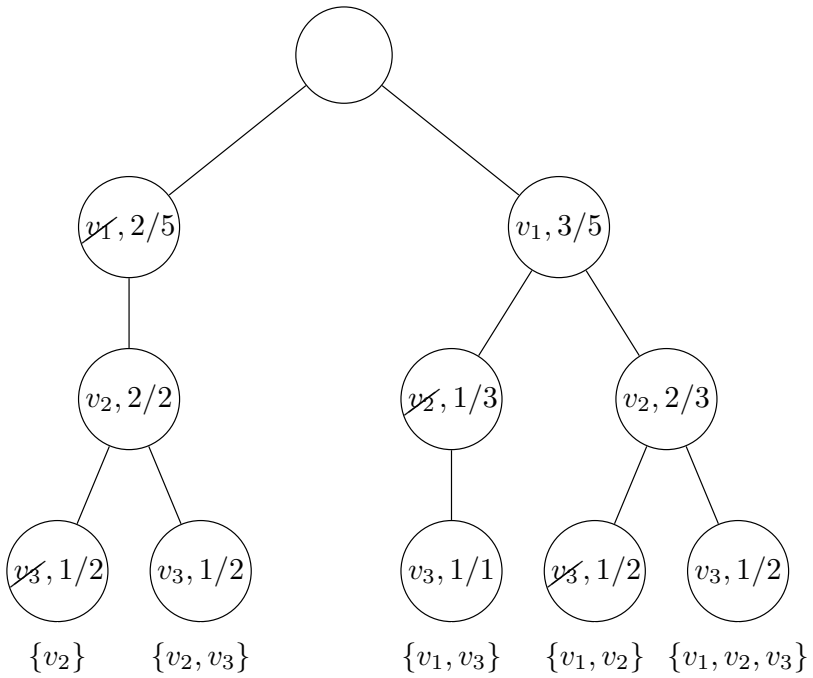


The Importance Sampling

Choose:

$$\tilde{p}_i^* = \frac{|\{\text{valid covers which contain } \mathbf{C} \cup \{v_i\}\}|}{|\{\text{valid covers which contain } \mathbf{C}\}|} \quad \forall i = 1, \ldots, n.$$





For example, suppose we generate $v_1 \to v_2 \to \cancel{v_3}$. In this case,

$$X = \tilde{p}_1^{-1} \cdot \tilde{p}_2^{-1} \cdot \tilde{q}_3^{-1} = \frac{5}{3} \cdot \frac{3}{2} \cdot \frac{2}{1} = 5.$$

Our problem summary

- Unfortunately, calculating

$$\tilde{p}_i^* = \frac{|\{\text{valid covers which contain } \mathbf{C} \cup \{v_i\}\}|}{|\{\text{valid covers which contain } \mathbf{C}\}|} \quad \forall i = 1, \ldots, n,$$

is hard. To find $\tilde{p}_i^*$, we have to calculate the number of vertex covers in G; but, this is the problem we tried to solve in the first place...

Our problem summary

- Unfortunately, calculating

$$\tilde{p}_i^* = \frac{|\{\text{valid covers which contain } \mathbf{C} \cup \{v_i\}\}|}{|\{\text{valid covers which contain } \mathbf{C}\}|} \quad \forall i = 1, \ldots, n,$$

  is hard. To find $\tilde{p}_i^*$, we have to calculate the number of vertex covers in G; but, this is the problem we tried to solve in the first place...
- Can we choose $\tilde{p}_i = 1/2$? – Yes, **BUT** the estimator will generally result in high variance...

Our problem summary

- Unfortunately, calculating

$$\tilde{p}_i^* = \frac{|\{\text{valid covers which contain } \mathbf{C} \cup \{v_i\}\}|}{|\{\text{valid covers which contain } \mathbf{C}\}|} \quad \forall i = 1, \ldots, n,$$

  is hard. To find $\tilde{p}_i^*$, we have to calculate the number of vertex covers in G; but, this is the problem we tried to solve in the first place...
- Can we choose $\tilde{p}_i = 1/2$? – Yes, **BUT** the estimator will generally result in high variance...
- This is a very important problem, and we will try to solve it using stochastic graph relaxation technique.

Stochastic Graph Relaxation (1)

- Let $G = (V, E)$ be a graph and let $|V| = n$. Introduce some vertex ordering $(v_1, v_2, \cdots v_n)$ and denote by $d_i = \{j \mid (v_i, v_j) \in E,\ j > i\}$ a neighbors set of v_i such that each neighbor v_j satisfies $j > i$.

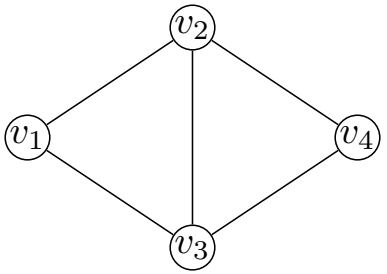
Stochastic Graph Relaxation (1)

- Let $G = (V, E)$ be a graph and let $|V| = n$. Introduce some vertex ordering $(v_1, v_2, \cdots v_n)$ and denote by $d_i = \{j \mid (v_i, v_j) \in E,\ j > i\}$ a neighbors set of v_i such that each neighbor v_j satisfies $j > i$.

Definition (Induced probability vector)

A probability vector induced by G is given by

$$P = (p_1, \cdots, p_n) = \left(\frac{|d_1|}{n-1}, \frac{|d_2|}{n-2}, \cdots, \frac{|d_{n-1}|}{1}, 0 \right).$$



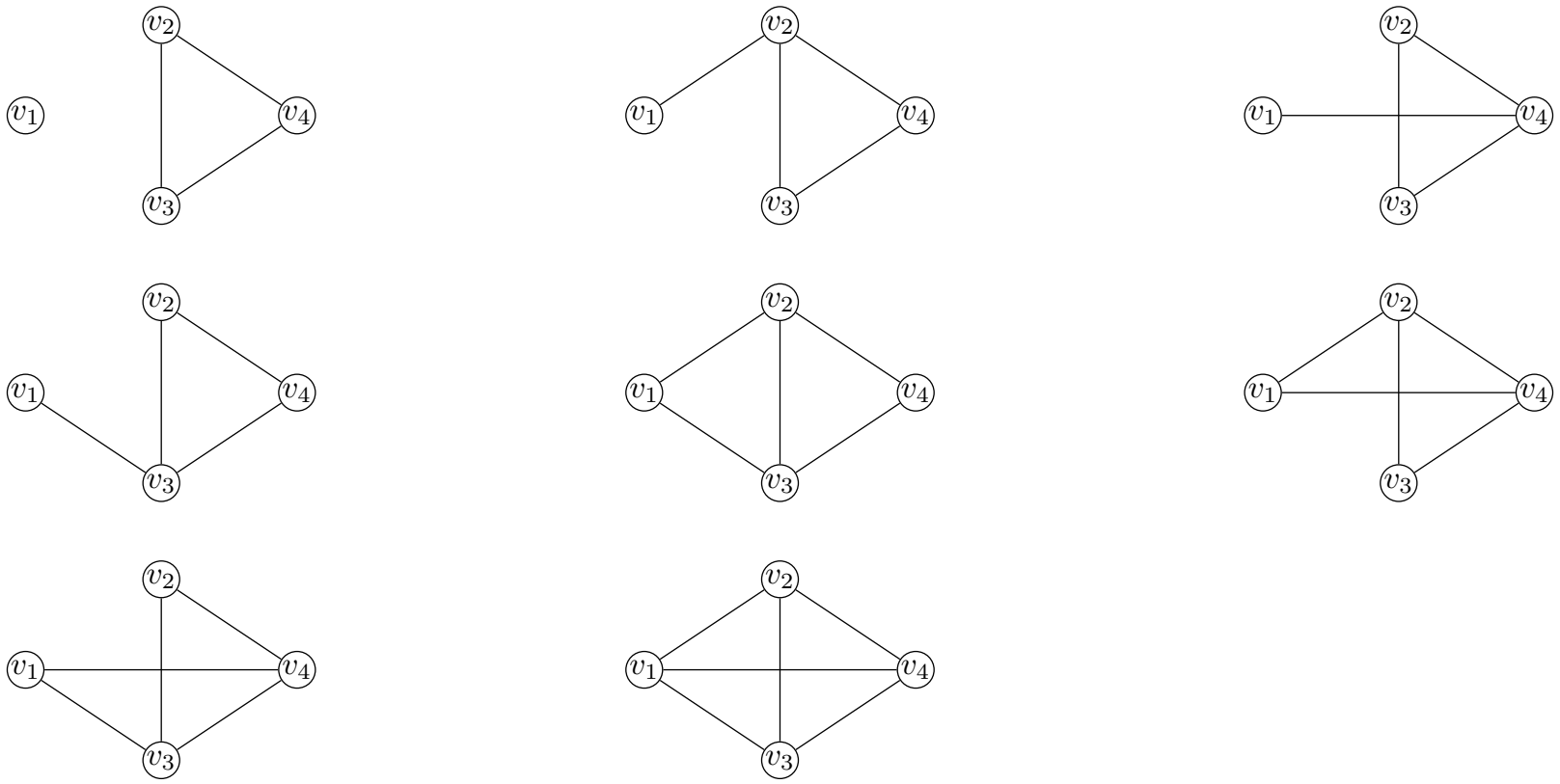
$$(v_1, v_2, v_3, v_4) \quad \Rightarrow \quad P = \left(\tfrac{2}{3}, \tfrac{2}{2}, \tfrac{1}{1}, 0\right).$$

Stochastic Graph Relaxation (2)

Consider a probability space Ω_G of all random graphs $G' = (V', E')$ where the set of vertexes remains the same as in G i.e $V' = V$ but each edge (v_i, v_j) $i < j$ is present with probability $p_i = \frac{|d_i|}{n-i}$. In particular, the follwing graphs can be generated from $P = \left(\frac{2}{3}, \frac{2}{2}, \frac{1}{1}, 0\right)$.

Stochastic Graph Relaxation (2)

Consider a probability space Ω_G of all random graphs $G' = (V', E')$ where the set of vertexes remains the same as in G i.e $V' = V$ but each edge (v_i, v_j) $i < j$ is present with probability $p_i = \frac{|d_i|}{n-i}$. In particular, the follwing graphs can be generated from $P = \left(\frac{2}{3}, \frac{2}{2}, \frac{1}{1}, 0\right)$.



Stochastic Graph Relaxation (3)

Let G' be a random variable such that $G' \in \Omega_G$, and note that $\mathbb{P}(G' \in \Omega_G)$ is well defined by probability vector P we can write the expected number of vertex covers under Ω_G as

$$\mathbb{E}_{\Omega_G}\left(|\mathrm{VC}(G')|\right) = \sum_{G'' \in \Omega_G} \mathbb{P}(G'')|\mathrm{VC}(G')|.$$

Stochastic Graph Relaxation (3)

Let G' be a random variable such that $G' \in \Omega_G$, and note that $\mathbb{P}(G' \in \Omega_G)$ is well defined by probability vector P we can write the expected number of vertex covers under Ω_G as

$$\mathbb{E}_{\Omega_G}\left(|\mathrm{VC}(G')|\right) = \sum_{G'' \in \Omega_G} \mathbb{P}(G'')|\mathrm{VC}(G')|.$$

Proposed approximation to the number of covers in G

We propose to use the $\mathbb{E}_{\Omega_G}\left(|\mathrm{VC}(G')|\right)$ value, as an approximation to the real number of covers in the original graph G.

Stochastic Graph Relaxation (3)

Let G' be a random variable such that $G' \in \Omega_G$, and note that $\mathbb{P}(G' \in \Omega_G)$ is well defined by probability vector P we can write the expected number of vertex covers under Ω_G as

$$\mathbb{E}_{\Omega_G}\left(|\mathrm{VC}(G')|\right) = \sum_{G'' \in \Omega_G} \mathbb{P}(G'')|\mathrm{VC}(G')|.$$

Proposed approximation to the number of covers in G

We propose to use the $\mathbb{E}_{\Omega_G}\left(|\mathrm{VC}(G')|\right)$ value, as an approximation to the real number of covers in the original graph G.

Theorem

There exists a deterministic polynomial time Dynamic Programming Algorithm that calculates $\mathbb{E}_{\Omega_G}\left(|\mathrm{VC}(G')|\right)$ *analytically.*

A nice feature — an exact vertex cover computation

Lemma

Given that an instance $G = (V, E)$ induce a probability vector $P = (p_1, \cdots, p_n)$ where each p_i satisfies $p_i \in \{0, 1\}$, the Dynamic Programming Algorithm provides the exact answer to the number of vertex covers, i.e $|\mathrm{VC}(G)| = \mathbb{E}_{\Omega_G}[|\mathrm{VC}(G')|]$.

A nice feature — an exact vertex cover computation

Lemma

Given that an instance $G = (V, E)$ induce a probability vector $P = (p_1, \cdots, p_n)$ where each p_i satisfies $p_i \in \{0, 1\}$, the Dynamic Programming Algorithm provides the exact answer to the number of vertex covers, i.e $|\mathrm{VC}(G)| = \mathbb{E}_{\Omega_G}[|\mathrm{VC}(G')|]$.

Proof.

Notice that there is only one graph in Ω_G when G induce a vector P where $p_i \in \{0, 1\}$, this follows immediately from the construction process of random graph under this particular Ω_G. In other words, we have that $\forall G'(V, E') \in \Omega_G,\ G'(V, E') = G(V, E)$ so $|\mathrm{VC}(G')| = \mathbb{E}_{\Omega_G}(|\mathrm{VC}(G')|)$.

$\square$

Random Graph $G(n, 1/2)$

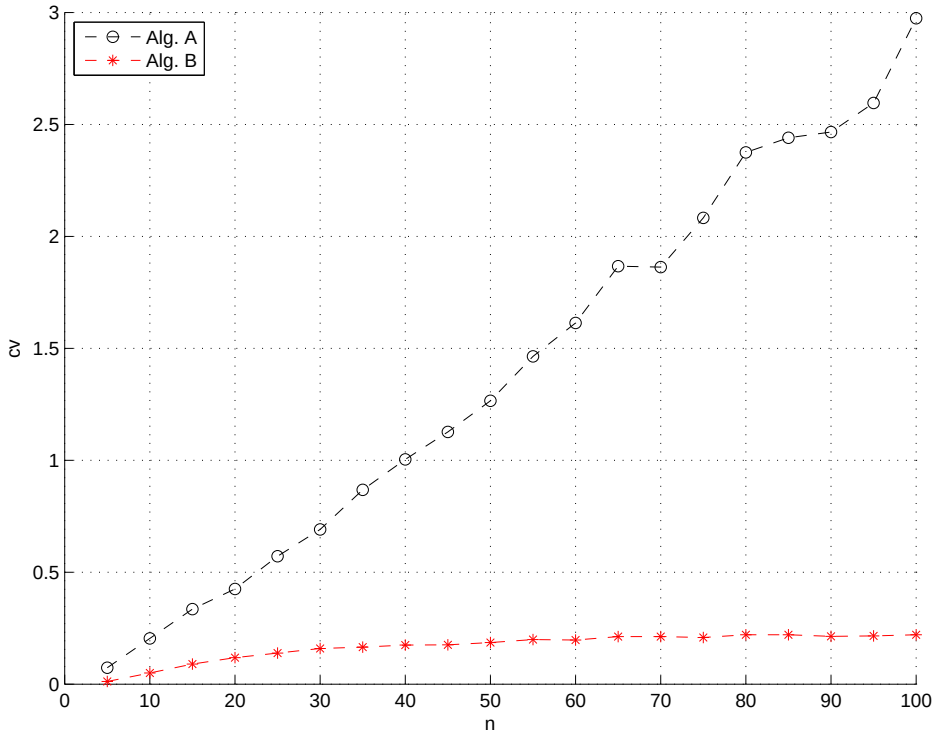


Figure: Algorithm A — Rassmussen Fully Polynomial Randomized Approximation Scheme. Algorithm B — SIS.

Model 1

Model 1 A graph with $|V| = 100$ and $|E| = 2,432$. The graph was generated in the following way. We defined the number of vertexes to be 100 and each edge (v_i, v_j) was generated with probability $Ber(p)$ while p is also a random variable such that $p \sim Uni(0, 1)$.

- *cachet* delivers an exact solution of $244,941$ in 0.75 seconds.
- We ran *SampleSearch* for 10 times and it provides an average of $192,251.25$ using 60 seconds time limit.
- The SIS Algorithm $(N = 100)$ delivered 2.440×10^5 in 1.698 seconds. The RE is 1.614×10^{-2}

The following figure provides a typical Histogram of the Importance Weights obtained in a single run of SIS Algorithm.

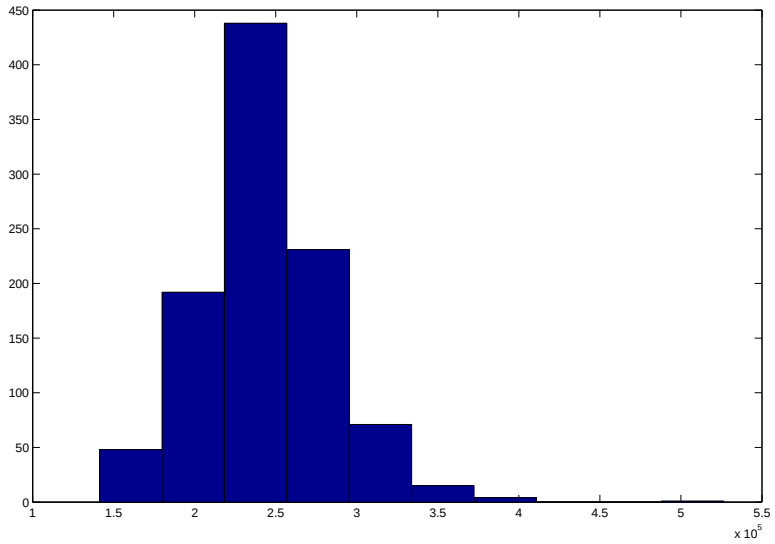


Figure: Histogram of 1,000 Importance Weights for **Model 1**.

Model 2

A graph with $|V| = 300$ and $|E| = 21,094$. The graph was generated in the following way. We defined the number of vertexes to be 300 and each edge (v_i, v_j) was generated with probability $Ber(p_i)$ while p_i is also a random variable such that $p \sim Uni(0,1)$. The results are self explanatory.

- *cachet* delivers an exact solution of 1.306×10^{14} in about 17 minutes.
- We ran *SampleSearch* for 10 times and it provides an average of 6.001×10^{13} using $1,200$ seconds time limit.
- The SIS Algorithm ($N = 100$) delivered 1.387×10^{14} in 56.64 seconds. The RE is 4.171×10^{-2}

The following figure provides a typical Histogram of the Importance Weights obtained in a single run of SIS Algorithm.

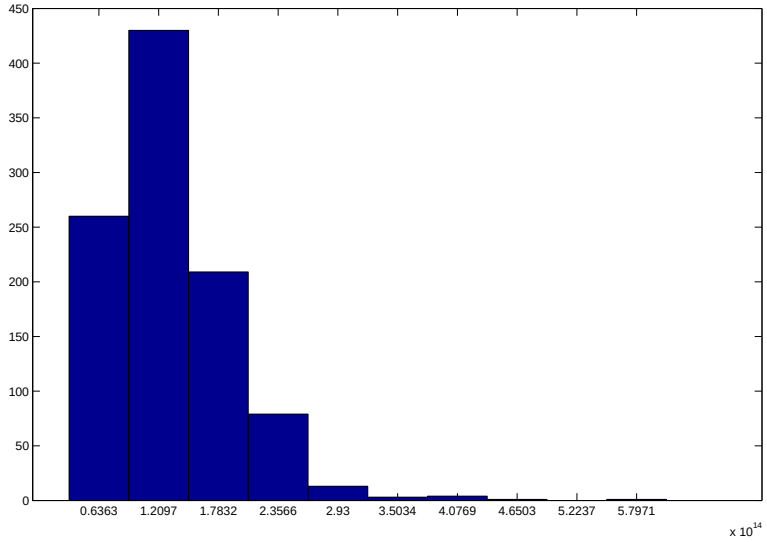


Figure: Histogram of 1,000 Importance Weights for **Model 2**.

Model 3

A graph with $|V| = 1,000$ and $|E| = 64,251$. The graph was generated in the following way. We defined the number of vertexes to be 1,000 and each edge (v_i, v_j) was generated from $Ber(p)$ while each p is generated from truncated Normal distribution with $\mu = 0.1$ and $\sigma = 0.1$ The results are summarized bellow.

- *cachet* was timed out after 2 days and was unable to deliver a solution. The lower bound of $3.439E + 09$ was supplied.
- *SampleSearch* failed to initialize, probably, because the problem is too big.
- The SIS Algorithm ($N = 100$) delivered 4.261×10^{32} in 648.6 seconds. The RE is 4.813×10^{-2}

The following figure provides a typical Histogram of the Importance Weights obtained in a single run of SIS Algorithm.

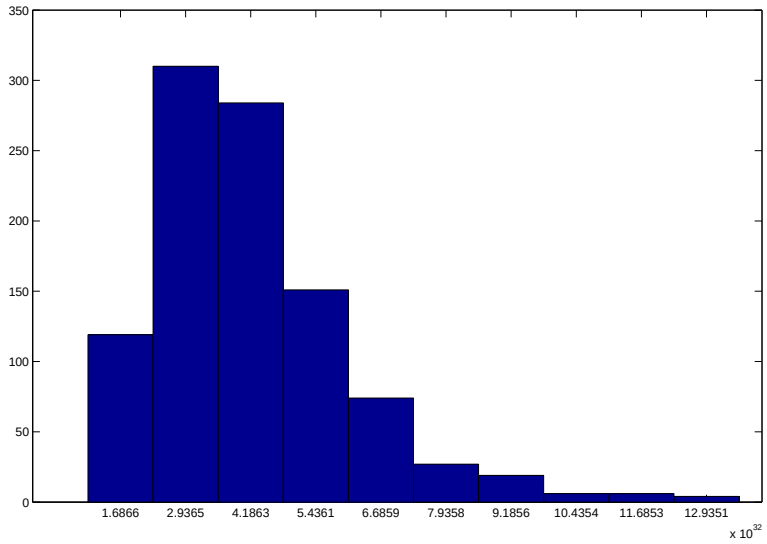


Figure: Histogram of 1,000 Importance Weights for **Model 3**.

Model 4

A graph with $|V| = 1,000$ and $|E| = 249,870$. The graph was generated in the following way. We defined the number of vertexes to be 1,000 and each edge (v_i, v_j) was generated from $Ber(p)$ while each p is generated from truncated Normal distribution with $\mu = 0.5$ and $\sigma = 0.3$ The results are summarized bellow.

- *cachet* was timed out after 2 days and was unable to deliver a solution. The lower bound of $9.601E + 10$ was supplied.
- *SampleSearch* failed to initialize, probably, because the problem is too big.
- The SIS Algorithm ($N = 100$) delivered 2.773×10^{11} in 1,718 seconds. The RE is 1.579×10^{-2}

The following figure provides a typical Histogram of the Importance Weights obtained in a single run of SIS Algorithm.

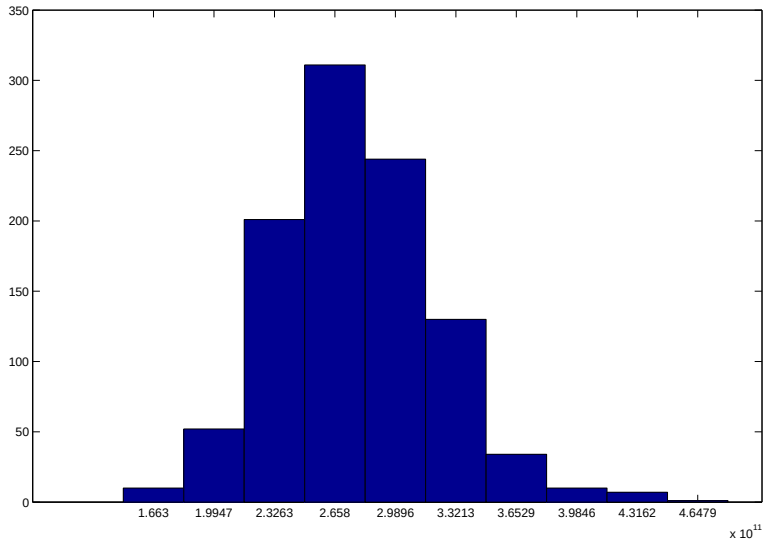


Figure: Histogram of 1,000 Importance Weights for **Model 4**.

What next?

- Extend the stochastic relaxation idea to other sampling problems.

What next?

- Extend the stochastic relaxation idea to other sampling problems.
- Prove the efficiency of this estimator for different families of graphs.

What next?

- Extend the stochastic relaxation idea to other sampling problems.
- Prove the efficiency of this estimator for different families of graphs.

Thank You!