Sequential Monte Carlo for Counting Vertex Covers in General Graphs

Radislav Vaisman[a], Zdravko I. Botev[b], Ad Ridder[c]

[a] School of Mathematics and Physics,
University of Queensland , Brisbane, Australia
r.vaisman@uq.edu.au

[b] The University of New South Wales,
Sydney, NSW 2052, Australia
botev@unsw.edu.au

[c] Faculty of Economics and Business Administration,
Vrije University, Amsterdam, The Netherlands
ad.ridder@vu.nl

August 27, 2014

Abstract

In this paper we describe a Sequential Importance Sampling (SIS) procedure for counting the number of vertex covers in general graphs. The optimal SIS proposal distribution is the uniform over a suitably restricted set, but is not implementable. We will consider two proposal distributions as approximations to the optimal. Both proposals are based on randomization techniques. The first randomization is the classic probability model of random graphs, and in fact, the resulting SIS algorithm shows polynomial

complexity (FPRAS) for random graphs. The second randomization introduces a probabilistic relaxation technique that uses Dynamic Programming. The numerical experiments show that the resulting SIS algorithm enjoys excellent practical performance in comparison with existing methods. In particular the method is compared with *cachet* - an exact model counter, and the state of the art *SampleSearch*, which is based on Belief Networks and importance sampling.

Keywords. Vertex Cover, Counting problem, Sequential importance sampling, Dynamic Programming, Relaxation, Random Graphs.

1 Introduction

In this article we propose an approximation algorithm for the vertex cover counting problem in a graph, by using Monte Carlo methods. In graph theory, 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.

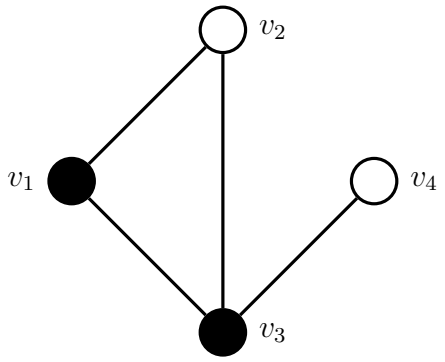

Figure 1: Example of vertex cover formed by v_1 and v_3.

The vertex cover problem is one of the classical NP-complete decision problems in computational complexity theory, see Karp [16]. Hence, it is often used for developing approximation algorithms in the domain of scientific computing. A natural generalization is the associated counting problem: how many vertex covers are there for a given graph? This is closely related to propositional model counting where it is asked to count the number of assignments for a given propositional formula [11]. Efficient model counting algorithms are of interest for Bayesian inference problems or combinatorial design problems.

Vertex cover counting is #P-complete, meaning—basically similar to NP-complete—that it is in #P, and that every #P problem can be reduced to it in polynomial counting reduction. We refer to Valiant [28] for more details on these complexity classes. Specifically, #P-completeness means that exact solutions of the counting problem cannot be obtained by polynomial-time algorithms.

Moreover, it has been shown in [9, 27] that counting the number of vertex covers remains hard even when restricted to planar bipartite graphs of bounded degree or regular graphs of constant degree. In addition, Liu and Lu [19] showed that there is no efficient approximation algorithm capable to count vertex covers if some vertex can appear in 6 clauses unless NP=RP (RP is the class of problems for which there is a randomized algorithm such that a YES answer is right with certainty but a NO answer is right with probability; RP is a subset of NP).

The above findings motivate to search for Monte Carlo approximation algorithms. Generally, most successful approximation algorithms of #P-complete counting problems are randomized schemes. To be more specific, suppose that $\ell(n)$ is the exact counting number of some instance of the problem of size n; for instance, $\ell(n)$ is the number of vertex covers of a graph $G = G(V, E)$ with $n = |V|$ vertices. For convenience we do not include the instance G in the notation, but clearly, $\ell(n)$ also depends on it. A *randomized algorithm* produces a random output $\widehat{\ell}(n)$ as estimate of $\ell(n)$, for instance by a Monte Carlo simulation method. A randomized algorithm is an (ε, δ)-*approximation algorithm* if

$$\mathbb{P}\big((1-\varepsilon)\ell(n) < \widehat{\ell}(n) < (1+\varepsilon)\ell(n)\big) > 1-\delta$$

for all $0 < \delta, \varepsilon < 1$. We allow ε and δ to be part of the input of the estimator, which we might denote by $\widehat{\ell}(n, \varepsilon, \delta)$. Then, a randomized algorithm is a *randomized approximation scheme* (RAS) if for every triple (n, ε, δ) the output satifies

$$\mathbb{P}\big((1-\varepsilon)\ell(n) < \widehat{\ell}(n, \varepsilon, \delta) < (1+\varepsilon)\ell(n)\big) > 1-\delta.$$

Finally, a RAS is called fully polynomial if its running time is polynomial in ε^{-1}, $\log \delta^{-1}$ and n. This concept of *fully polynomial randomized approximation*

scheme (FPRAS) was introduced in Karp and Luby [17]. For example, FPRAS has been successfully developed in

- Karp et al. [18] for counting the number of satisfying assignments to a boolean formula in disjunctive normal form.
- Jerrum and Sinclair [13] for counting the number of matchings (of all sizes) in a graph.
- Cryan and Dyer [6] for the number of contingency tables when the number of rows is constant.
- Jerrum et al. [14] for counting the permanent of a matrix with nonnegative entries.
- Dyer [8] for counting the number of solutions to a 0-1 knapsack problem.

There are two Monte Carlo statistical approaches to tackle these difficult counting problems. The first is *Markov Chain Monte Carlo* (MCMC) and the second is *sequential importance sampling* (SIS). Both approaches exploit the finding of Jerrum et al. [15] that counting is equivalent to uniform sampling over a suitably restricted set.

MCMC methods sample from such restricted regions by constructing an ergodic Markov Chain with stationary and limiting distribution equal to the desired uniform distribution. A number of MCMC approaches with good empirical performance have been proposed [4, 13, 22, 24] for counting problems such as satisfiability, knapsack, matching, permanent, but to our knowledge not for vertex cover.

In this article we focus on the SIS approach in much the same spirit as Rasmussen [21] and Chen at al. [5]. In addition to [5], there are many examples of

successful SIS implementations on various counting problems, see, for example, [2, 17, 23]. The motivation of this article is to find a successful application of SIS to yet another important counting problem — counting the number of vertex covers of a graph.

The rest of the paper is organized as follows. Section 2 describes the importance sampling method for approximate counting, specifically the sequential version of importance sampling (SIS) and its associated algorithm for vertex cover counting. A large body of this section is devoted to derive the exact expressions of the optimal proposal distribution for executing SIS, also known as the zero-variance proposal distribution. In Section 3 we present our first algoritm for executing SIS using a proposal distribution that approximates the zero-variance. The approximation is based on the classic probabilistic model of random graphs. The conditional distributions of the proposal are computed using the expected number of vertex covers of specific random subgraphs in this probability model. We then show how the resulting SIS Algorithm A corresponds with a randomized algorithm for counting cliques developed in [21]. As a consequence, Algorithm A shows FPRAS for random graphs. Section 4 considers a new randomization technique for constructing a proposal SIS distribution as approximation of the zero-variance. The randomization is based on a probabilistic relaxation to the vertex cover problem. The expected number of vertex covers of random graphs in this probability model is computed efficiently by dynamic programming. The resulting SIS Algorithm B is shown in Section 5 to outperform Algorithm A in terms of statistical properties of the associated estimators. Also, we provide numerical support for the accuracy of our method by comparing its performance with existing procedures. Overall, the proposed method is more efficient than available alternatives. Finally, in Section 6 we summarize our findings and dis-

cuss possible directions for future research.

2 Importance Sampling

Given is an undirected graph $G = G(V, E)$, with vertex set $|V| = n$ consisting of n labeled vertices $v_1, \ldots, v_n$, and edge set E. We consider *simple graphs* only: no loops and no more than one edge between different vertices. Denote by $\mathcal{X}_G$ the set of all vertex covers of the given graph G. The goal is to compute its cardinality $|\mathcal{X}_G|$.

Consider the binary n-space $\{0,1\}^n$ of all binary n-tuples $\boldsymbol{x} = (x_1, \ldots, x_n),\ x_i \in \{0,1\}$. With any n-tuple $\boldsymbol{x} \in \{0,1\}^n$ we associate a subset $V(\boldsymbol{x}) \subset V$ of vertices by letting $v_i \in V(\boldsymbol{x})$ if and only if $x_i = 1$. Clearly, this subset may or may not be a vertex cover of the given graph G. Thus we can say that

$$|\mathcal{X}_G| = \sum_{\boldsymbol{x} \in \{0,1\}^n} \mathbb{I}\{V(\boldsymbol{x}) \in \mathcal{X}_G\}.$$

Suppose that we introduce a randomization on the binary n-space, meaning that n-tuple $\boldsymbol{x}$ is generated with probability $f(\boldsymbol{x})$, where $f(\boldsymbol{x}) \geq 0$ and $\sum_{\boldsymbol{x} \in \{0,1\}^n} f(\boldsymbol{x}) = 1$. We denote the random n-tuple by $\boldsymbol{X}$, the associated probabilities and expectations by $\mathbb{P}_f$ and $\mathbb{E}_f$, and we call $f(\cdot)$ a *probability mass function* (PMF).

In this section we consider estimating the number of vertex covers $|\mathcal{X}_G|$ by importance sampling simulations using a proposal PMF $f(\cdot)$ of the random n-tuple. For that purpose, we restrict the class of proposal PMF's by requiring positive probability of all n-tuples for which the associated vertex subset is a vertex cover:

$$\mathscr{F} = \Big\{ f : \{0,1\}^n \to [0,1];\ \sum_{\boldsymbol{x} \in \{0,1\}^n} f(\boldsymbol{x}) = 1;\ V(\boldsymbol{x}) \in \mathcal{X}_G \ \Rightarrow\ f(\boldsymbol{x}) > 0 \Big\}. \quad (1)$$

Using a proposal PMF $f \in \mathscr{F}$, the corresponding single-run importance sampling estimator is

$$Z_f = \frac{\mathbb{I}\{V(\boldsymbol{X}) \in \mathcal{X}_G\}}{f(\boldsymbol{X})}. \tag{2}$$

Clearly, this estimator is unbiased:

$$\mathbb{E}_f[Z_f] = \mathbb{E}_f\left[\frac{\mathbb{I}\{V(\boldsymbol{X}) \in \mathcal{X}_G\}}{f(\boldsymbol{X})}\right] = \sum_{\boldsymbol{x}\in\{0,1\}^n : V(\boldsymbol{x})\in\mathcal{X}_G} \frac{1}{f(\boldsymbol{x})} \times f(\boldsymbol{x}) = |\mathcal{X}_G| \,.$$

This identity suggests the *importance sampling estimator* by running the algorithm independently many times:

$$\widehat{|\mathcal{X}_G|} = \frac{1}{N}\sum_{i=1}^{N} Z_f^{(i)} = \frac{1}{N}\sum_{i=1}^{N} \frac{\mathbb{I}\{V(\boldsymbol{X}_i) \in \mathcal{X}_G\}}{f(\boldsymbol{X}_i)} \,, \tag{3}$$

where $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N$ are IID random n-tuples generated by PMF $f(\cdot)$, and $V(\boldsymbol{X}_1)$, $\ldots, V(\boldsymbol{X}_N)$ are their associated (random) vertex subsets.

A measure of efficiency of an estimator is its *coefficient of variation* (CV) defined as the ratio of the variance to the square of the first moment [5]. Hence, the CV of the single-run estimator is

$$\mathrm{cv}^2 = \frac{\mathbb{V}\mathrm{ar}[Z_f]}{|\mathcal{X}_G|^2},$$

which is estimated simply by

$$\widehat{\mathrm{cv}^2} = \frac{\frac{1}{N-1}\sum_{i=1}^{N}\left(Z_f^{(i)} - \widehat{|\mathcal{X}_G|}\right)^2}{\widehat{|\mathcal{X}_G|}^2} \,.$$

By Slutsky's theorem it follows that $\widehat{\mathrm{cv}^2}$ is asymptotically consistent. The CV of the importance sampling estimator $\widehat{|\mathcal{X}_G|}$ equals cv^2/N. The square root of the

CV is commonly known as the *relative error* (RE). In our numerical tests we will estimate RE by

$$\widehat{\mathrm{RE}} = \widehat{\mathrm{cv}}/\sqrt{N}. \tag{4}$$

2.1 Sequential Importance Sampling

An importance sampling simulation is implemented more efficiently by a sequential procedure. We now proceed with the details of a sequential importance sampling simulation (SIS) procedure for the vertex cover problem. First, recall that $f(\boldsymbol{x})$ can be decomposed as a product of conditional PMF's:

$$f(\boldsymbol{x}) = f_1(x_1) f_2(x_2|x_1) \dots f_n(x_n|x_1, \dots, x_{n-1}) \,, \tag{5}$$

where

$$f_i(x_i|x_1, \cdots, x_{i-1}) = \frac{\sum_{x_{i+1}\in\{0,1\},\dots,x_n\in\{0,1\}} f(x_1, \dots, x_i, x_{i+1}, \dots, x_n)}{\sum_{x_i\in\{0,1\},\dots,x_n\in\{0,1\}} f(x_1, \dots, x_{i-1}, x_i, \dots, x_n)}$$

This decomposition allows us to sample vertex subsets in a sequential manner, ensuring that only valid vertex covers are sampled. In particular, suppose we start adding vertices v_i for $i \in \{1, 2, \cdots\}$ to the vertex cover one by one with probability $f_i(x_i|x_1, \cdots, x_{i-1})$ and consider step i. We start this step with vertices $v_j, j < i$ for which $x_j = 1$ form a vertex cover in the subgraph induced by all vertices $v_1, \dots, v_{i-1}$.

We either add v_i to the vertex cover, or we do not. While adding v_i to the cover is always feasible, not adding v_i is only feasible if there is no vertex v_j, $j < i$, that is not chosen ($x_j = 0$) and that is a neigbor of v_i ($(v_j, v_i) \in E$). This leads to the following SIS algorithm.

Algorithm 2.1 ***Sequential Sampling of Valid Covers***

Input: Graph $G = G(V, E)$ *and proposal PMF* $f(\cdot)$.

Output: Importance weight of the generated vertex cover.

1: $Z \leftarrow 1$, $\boldsymbol{x} \leftarrow (0, \ldots, 0)$, *so that* $V(\boldsymbol{x}) = \emptyset$.

2: ***for*** $i = 1 \rightarrow n$ ***do*** ▷ *Note that at stage* i, $V(\boldsymbol{x}) \subseteq \{v_1, \ldots, v_{i-1}\}$

3: ***if*** v_i *must be added to the cover* ***then***

4: $x_i \leftarrow 1$

5: ***else***

6: $U \sim \mathsf{U}(0, 1)$

7: ***if*** $U \leq f_i(1|x_1, \ldots, x_{i-1})$ ***then***

8: $x_i \leftarrow 1$, $Z \leftarrow Z \times \frac{1}{f_i(1|x_1,\ldots,x_{i-1})}$

9: ***else***

10: $Z \leftarrow Z \times \frac{1}{f_i(0|x_1,\ldots,x_{i-1})}$

11: ***end if***

12: ***end if***

13: ***end for***

14: $Z \leftarrow Z \times \mathbb{I}\{V(\boldsymbol{x}) \in \mathcal{X}_G\}$

15: ***return*** *Repeat the above procedure* N *times to generate* $Z_1, \ldots, Z_N$, *and deliver the average* $\frac{1}{N}\sum_{i=1}^{N} Z_i$.

2.2 Zero-Variance Importance Sampling

The main issue in importance sampling simulations is to construct a good or optimal *change of measure*. In our setting this means to use an optimal PMF $f(\cdot)$ for generating the binary n-tuples $\boldsymbol{x}$. Clearly, optimal would be that the associated importance sampling estimator has *zero variance*. It suffices that this hold for the single-run importance sampling estimator; i.e., a PMF $f(\cdot)$ for which

$\mathbb{V}\mathrm{ar}[Z_f] = 0$ is called a *zero-variance PMF*, and it is an optimal importance sampling PMF, see [1, 25].

Miraculously, we can obtain such a zero-variance PMF. The next lemma shows that the uniform distribution on the space of vertex covers is the zero-variance pmf.

Lemma 2.1 *The PMF*

$$f^*(\boldsymbol{x}) = \frac{1}{|\mathcal{X}_G|}\,\mathbb{I}\{V(\boldsymbol{x}) \in \mathcal{X}_G\} \tag{6}$$

is a zero-variance PMF.

Proof. Clearly $f^* \in \mathscr{F}$, thus f^* is a feasible PMF. The associated importance sampling estimator Z_{f^*} is by (2)

$$Z_{f^*} = |\mathcal{X}_G|\,\mathbb{I}\{V(\boldsymbol{x}) \in \mathcal{X}_G\}$$

Hence, the second moment equals $|\mathcal{X}_G|^2$. For additional information about zero-variance PMF's, see [1, 25]. □

Because we execute the importance sampling simulation through the implementation of the sequential procedure of Section 2.1, we will analyse also the conditional PMF's of the zero-variance distribution. For that purpose, we need to introduce subgraphs $G^{[i]}$ and $G^{[-i]}$ $(i = 1, \ldots, n-1)$.

Definition 2.1 *Let be given binary $x_1, \ldots, x_{i-1}$ indicating whether nodes $v_j, j < i$, are part of a vertex subset $V(\boldsymbol{x})$.*

1. $i = 1$.

- $G^{[1]} = G(V_1, E_1)$ *where* $V_1 = \{v_2, \ldots, v_n\}$, *and* $E_1 = \{(v_j, v_k) | v_j, v_k \in V_1\} \cap E$;
- $G^{[-1]} = G(V_2, E_2)$ *where* $V_2 = V_1 \setminus \{v_k | k \geq 2, (v_1, v_k) \in E\}$, *and* $E_2 = \{(v_j, v_k) | v_j, v_k \in V_2\} \cap E$.

2. $i = 2, \ldots, n-1$.

- $G^{[i]} = G(V_1, E_1)$ *where*

$$V_1 = \{v_{i+1}, \ldots, v_n\} \setminus \{v_k | k \geq i+1, \text{and } \exists\, j \leq i-1 \text{with } x_j = 0, (v_j, v_k) \in E\}, \tag{7}$$

and $E_1 = \{(v_j, v_k) | v_j, v_k \in V_1\} \cap E$;

- $G^{[-i]} = G(V_2, E_2)$ *where*

$$V_2 = V_1 \setminus \{v_k | k \geq i+1, (v_i, v_k) \in E\}, \tag{8}$$

and $E_2 = \{(v_j, v_k) | v_j, v_k \in V_2\} \cap E$.

Note that these subgraphs depend on the given variables $x_1, \ldots, x_{i-1}$. For convenience we do not denote this explicitly. Note also that a subgraph can be the empty set.

Example 2.1 *Consider the bridge graph depicted in Figure 2.*

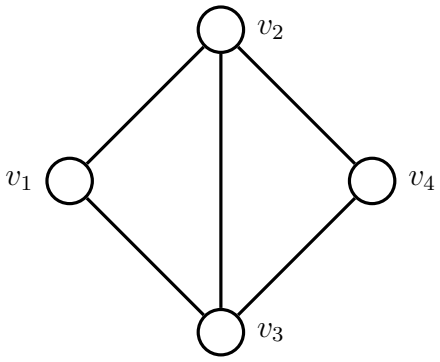

Figure 2: A bridge graph.

The $G^{[1]}$ and $G^{[-1]}$ graphs are depicted in Figure 3.

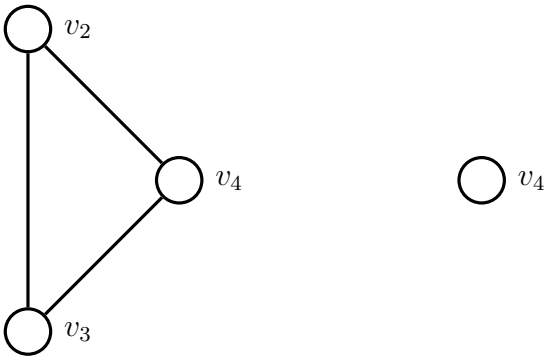

Figure 3: Left panel: $G^{[1]}$ graph. Right panel: $G^{[-1]}$ graph.

Consider $x_1 = 1$ and $i = 2$. Then,

- *$G^{[2]}$ has vertex set $V_1 = \{v_3, v_4\}$ and edge set $E_2 = \{(v_3, v_4)\}$.*
- *$G^{[-2]}$ has vertex set $V_2 = V_1 \setminus \{v_3, v_4\} = \emptyset$.*

Consider $x_1 = 1, x_2 = 1$ and $i = 3$. Then,

- *$G^{[3]}$ has vertex set $V_1 = \{v_4\}$.*
- *$G^{[-3]}$ has vertex set $V_2 = V_1 \setminus \{v_4\} = \emptyset$.*

□

Lemma 2.2 *Let be given binary variables $x_1, \ldots, x_{i-1}$, and denote the number of vertex covers in the associated $G^{[i]}$ and $G^{[-i]}$ graphs by $|\mathcal{X}_{G^{[i]}}|$ and $|\mathcal{X}_{G^{[-i]}}|$, where $|\mathcal{X}_{\emptyset}| = 1$. Then the zero-variance conditional PMF is as follows.*

(a). Case $i = 1$.

$$\begin{aligned} f_1^*(1) &= \frac{|\mathcal{X}_{G^{[1]}}|}{|\mathcal{X}_{G^{[1]}}| + |\mathcal{X}_{G^{[-1]}}|} \\ f_1^*(0) &= 1 - f_1^*(1) \end{aligned} \tag{9}$$

(b). Case $i = 2, \ldots, n$, and there is a node $v_j, j < i$, such that $x_j = 0$, and $(x_j, x_i) \in E$.

$$\begin{aligned} f_i^*(1|x_1, \ldots, x_{i-1}) &= 1 \\ f_i^*(0|x_1, \ldots, x_{i-1}) &= 0 \end{aligned}$$

(c). Case $i = 2, \ldots, n-1$, and for all nodes $v_j, j < i$, that have $x_j = 0$, it holds that $(x_j, x_i) \notin E$.

$$\begin{aligned} f_i^*(1|x_1, \ldots, x_{i-1}) &= \frac{|\mathcal{X}_{G^{[i]}}|}{|\mathcal{X}_{G^{[i]}}| + |\mathcal{X}_{G^{[-i]}}|} \\ f_i^*(0|x_1, \ldots, x_{i-1}) &= 1 - f_i^*(1|x_1, \ldots, x_{i-1}) \end{aligned} \tag{10}$$

(d). Case $i = n$, and for all nodes $v_j, j < n$, that have $x_j = 0$, it holds that $(x_j, x_n) \notin E$.

$$f_n^*(1|x_1, \ldots, x_{n-1}) = f_n^*(0|x_1, \ldots, x_{n-1}) = \frac{1}{2}$$

Proof. We elaborate on case (c). Case (a) follows similarly, while cases (b) and (d) are straightforward. Because the (unconditional) zero-variance PMF $f^*(\cdot)$ is the uniform distribution on the space $\mathcal{X}_G$ of vertex covers, we get for the conditional PMF

$$f_i^*(1|x_1,\ldots,x_{i-1}) = \frac{|\{V(\boldsymbol{y}) \in \mathcal{X}_G : y_1 = x_1,\ldots,y_{i-1} = x_{i-1}, y_i = 1\}|}{|\{V(\boldsymbol{y}) \in \mathcal{X}_G : y_1 = x_1,\ldots,y_{i-1} = x_{i-1}\}|}. \tag{11}$$

The variables $x_1,\ldots,x_{i-1}$ have assigned values 0 or 1 in such a manner that all edges $(v_j, v_k) \cap E, j, k \leq i-1$ are covered. Consider any node $v_k, k \geq i+1$, and suppose that there is a node $v_j, j \leq i-1$ such that $x_j = 0$ and that edge $(v_j, v_k) \in E$. This means that v_j is not part of the vertex cover. Hence, to cover the edge (v_j, v_k), node v_k is assigned $x_k = 1$ for certain.

- If we would include v_i in the cover by setting $x_i = 1$, we obtain the subgraph $G^{[i]}$ with vertex set $V_1 \subset \{v_{i+1},\ldots,v_n\}$ given in (7). Clearly, we can map the vertex covers of $G^{[i]}$ one-to-one on those covers of G that are given by the variables $x_1,\ldots,x_{i-1}$ and $x_i = 1$. This means that,

$$|\{V(\boldsymbol{y}) \in \mathcal{X}_G : y_1 = x_1,\ldots,y_{i-1} = x_{i-1}, y_i = 1\}| = |\mathcal{X}_{G^{[i]}}|. \tag{12}$$

- Suppose that we do not include v_i in the cover, by setting $x_i = 0$. To cover an edge $(v_i, v_k) \in E, k \geq i+1$, node v_k is assigned $x_k = 1$ for sure. In this way we obtain the subgraph $G^{[-i]}$ with vertex set V_2 given in (8). Now we can map the vertex covers of $G^{[-i]}$ one-to-one on those covers of G that are given by the variables $x_1,\ldots,x_{i-1}$ and $x_i = 0$. That means that

$$|\{V(\boldsymbol{y}) \in \mathcal{X}_G : y_1 = x_1,\ldots,y_{i-1} = x_{i-1}, y_i = 0\}| = |\mathcal{X}_{G^{[-i]}}|. \tag{13}$$

The conclusion that (10) and (11) are equivalent, follows immediately. □

Example 2.2 *Consider the bridge example graph in Figure 2.*

1. $i = 1$. *The subgraphs* $G^{[1]}$ *and* $G^{[-1]}$ *are shown in Figure 3. Clearly we get* $|\mathcal{X}_{G^{[1]}}| = 4$ *and* $|\mathcal{X}_{G^{[-1]}}| = 2$. *Thus by (a) in Lemma 2.2,*

$$f_1^*(1) = \frac{4}{6} = \frac{2}{3}; \quad f_1^*(0) = \frac{1}{3}.$$

2. $i = 2$.

- *Case* $x_1 = 1$. *See Example 2.1 for* $G^{[2]} = \big(\{v_3, v_4\}, \{(v_3, v_4)\}\big)$ *and* $G^{[-2]} = \emptyset$. *Thus,* $|\mathcal{X}_{G^{[2]}}| = 3$, $|\mathcal{X}_{G^{[-2]}}| = 1$, *and by (c) in Lemma 2.2,*

$$f_2^*(1|1) = \frac{3}{4}; \quad f_2^*(0|1) = \frac{1}{4}.$$

- *Case* $x_1 = 0$. *By (b) in Lemma 2.2*

$$f_2^*(1|0) = 1; \quad f_2^*(0|1) = 0.$$

3. $i = 3$.

- *Case* $x_1 = 1, x_2 = 1$. *See Example 2.1 for* $G^{[3]} = \big(\{v_4\}, \emptyset\big)$ *and* $G^{[-3]} = \emptyset$. *Thus,* $|\mathcal{X}_{G^{[3]}}| = 2$, $|\mathcal{X}_{G^{[-3]}}| = 1$, *and by (c) in Lemma 2.2,*

$$f_3^*(1|1,1) = \frac{2}{3}; \quad f_3^*(0|1,1) = \frac{1}{3}.$$

- *Case* $x_1 = 1, x_2 = 0$. *By (b) in Lemma 2.2*

$$f_3^*(1|1,0) = 1; \quad f_3^*(0|1,0) = 0.$$

- *Case* $x_1 = 0, x_2 = 1$. *By (b) in Lemma 2.2*

$$f_3^*(1|0,1) = 1; \quad f_3^*(0|0,1) = 0.$$

4. $i = 4$.

- *Case* $x_1 = 1, x_2 = 1, x_3 = 1$. *By (d) in Lemma 2.2,*

$$f_4^*(1|1,1,1) = f_4^*(0|1,1,1) = \frac{1}{2}.$$

- *Case* $x_1 = 1, x_2 = 1, x_3 = 0$. *By (b) in Lemma 2.2*

$$f_4^*(1|1,1,0) = 1; \quad f_4^*(0|1,1,0) = 0.$$

- *Case* $x_1 = 1, x_2 = 0, x_3 = 1$. *By (b) in Lemma 2.2*

$$f_4^*(1|1,0,1) = 1; \quad f_4^*(0|1,0,1) = 0.$$

- *Case* $x_1 = 0, x_2 = 1, x_3 = 1$. *By (d) in Lemma 2.2,*

$$f_4^*(1|0,1,1) = f_4^*(0|0,1,1) = \frac{1}{2}.$$

Note that the product (5) *of these conditional PMF's indeed gives the uniform*

distribution on the space of all vertex covers:

$$
\begin{aligned}
f^*(1,1,1,1) &= f_1^*(1) f_2^*(1|1) f_3^*(1|1,1) f_4^*(1|1,1,1) = \frac{2}{3}\,\frac{3}{4}\,\frac{2}{3}\,\frac{1}{2} = \frac{1}{6} \\
f^*(1,1,1,0) &= f_1^*(1) f_2^*(1|1) f_3^*(1|1,1) f_4^*(0|1,1,1) = \frac{2}{3}\,\frac{3}{4}\,\frac{2}{3}\,\frac{1}{2} = \frac{1}{6} \\
f^*(1,1,0,1) &= f_1^*(1) f_2^*(1|1) f_3^*(0|1,1) f_4^*(1|1,1,0) = \frac{2}{3}\,\frac{3}{4}\,\frac{1}{3}\,1 = \frac{1}{6} \\
f^*(1,0,1,1) &= f_1^*(1) f_2^*(0|1) f_3^*(1|1,0) f_4^*(1|1,0,1) = \frac{2}{3}\,\frac{1}{4}\,1\,1 = \frac{1}{6} \\
f^*(0,1,1,1) &= f_1^*(0) f_2^*(1|0) f_3^*(1|0,1) f_4^*(1|0,1,1) = \frac{1}{3}\,1\,1\,\frac{1}{2} = \frac{1}{6} \\
f^*(0,1,1,0) &= f_1^*(0) f_2^*(1|0) f_3^*(1|0,1) f_4^*(0|0,1,1) = \frac{1}{3}\,1\,1\,\frac{1}{2} = \frac{1}{6}
\end{aligned}
$$

□

Since we can not calculate generally the zero-variance conditional PMF $f_i^*(x_i|x_1,\ldots,x_{i-1})$ exactly for the (a) and (c) cases in Lemma 2.2, we shall consider in the following sections two proposal PMF's $f(\cdot)$ as approximations of $f^*(\cdot)$. Recall that the ultimate goal is that the associated randomized scheme has polynomially complexity (FPRAS).

3 Algorithm A

Our first algorithm is inspired by a randomization idea. Recall that our given graph is $G = G(V,E)$ with $|V| = n$ vertices. In this section we shall consider next to vertex covers other subsets of the vertex set V. Therefore, $\mathcal{X}_G^{\text{vc}}$ denotes the set of vertex covers.

Definition 3.1 *For any subgraph $H \subset G$ containing k vertices we let*

$$E_k = \sum_{i=0}^{k} \binom{k}{i} 2^{-\binom{i}{2}}. \tag{14}$$

The quantity E_k has the following probabilistic interpretation. Suppose that we generate a random graph with k vertices according to the *Edgar Gilbert* model, see [3]. This means that each edge from the $\binom{k}{2}$ possible edges is present with probability $1/2$. We denote a random graph in this model by $\mathscr{G}$. The cardinality of the set of all vertex covers of the random graph is the random variable $|\mathcal{X}_{\mathscr{G}}|$. Probabilities and expectations in this random model are denoted by $\mathbb{P}_{\mathrm{EG}}$ and $\mathbb{E}_{\mathrm{EG}}$ to make the distinction with the probability model associated with the importance sampling PMF $f(\cdot)$ that we considered in the previous section. EG referd to the Edgar Gilbert model.

Lemma 3.1 *Let $\mathscr{G}$ be a random graph of k vertices. Then,*

$$\mathbb{E}_{\mathrm{EG}}\left[|\mathcal{X}_{\mathscr{G}}^{\mathrm{vc}}|\right] = E_k \ . \tag{15}$$

In words, E_k equals the expected number of vertex covers in the random graph.

Proof. Denote the number of vertex covers of size i (where $i = 0, \ldots, k$) by $|\mathcal{X}_{\mathscr{G}}^{\mathrm{vc}}(i)|$. Thus,

$$\mathbb{E}_{\mathrm{EG}}|\mathcal{X}_{\mathscr{G}}^{\mathrm{vc}}| = \sum_{i=0}^{k} \mathbb{E}_{\mathrm{EG}}|\mathcal{X}_{\mathscr{G}}^{\mathrm{vc}}(i)|.$$

Let $G = G(V, E)$ be an instance of the random graph. A subset $S \subset V$ of vertices is an *independent set* in G if no two vertices of S are adjacent. It is well known that a subset $T \subset V$ of vertices is a vertex cover if and only if $S = V \setminus T$ is an independent set. Thus in the random graph model we clearly have that

the expected number of vertex covers of size i equals the expected number of independent sets of size $k-i$:

$$\mathbb{E}_{\mathrm{EG}}|\mathcal{X}_{\mathscr{G}}^{\mathrm{vc}}(i)| = \mathbb{E}_{\mathrm{EG}}|\mathcal{X}_{\mathscr{G}}^{\mathrm{is}}(k-i)|.$$

However, the latter is easily to compute by reasoning that

(i). there are $\binom{k}{k-i}$ distinct subsets of size $k-i$;

(ii). a subset of size $k-i$ is independent with probability $2^{-\binom{k-i}{2}}$ as none of the edges between the $k-i$ vertices may be present.

Summarizing we get for the number of expected vertex covers in the random graph with k vertices

$$\begin{aligned}
\mathbb{E}_{\mathrm{EG}}|\mathcal{X}_{\mathscr{G}}^{\mathrm{vc}}| &= \sum_{i=0}^{k} \mathbb{E}_{\mathrm{EG}}|\mathcal{X}_{\mathscr{G}}^{\mathrm{vc}}(i)| \\
&= \sum_{i=0}^{k} \mathbb{E}_{\mathrm{EG}}|\mathcal{X}_{\mathscr{G}}^{\mathrm{is}}(k-i)| \\
&= \sum_{i=0}^{k} \binom{k}{k-i} 2^{-\binom{k-i}{2}} \\
&= \sum_{i=0}^{k} \binom{k}{i} 2^{-\binom{i}{2}} = E_k.
\end{aligned}$$

□

Our first randomized algorithm for counting vertex covers in a given graph $G = G(V, E)$ with $|V| = n$ vertices is the SIS Algorithm 2.1 with the following specially designed proposal PMF $f(\cdot)$. It suffices to specify the conditional PMF's $f_i(1|x_1, \ldots, x_{i-1})$ for any binary tuple $(x_1, \ldots, x_{i-1})$. Let be given such a tuple. Suppose that node v_i may or may not be included in the vertex cover (oth-

erwise $x_i = 1$ with probability 1). Then we determine the associated subgraphs $G^{[i]}$ and $G^{[-i]}$ from Definition 2.1. Let these graphs have k_+ and k_- vertices, respectively. Then we let

$$f_i(1|x_1, \ldots, x_{i-1}) = \frac{E_{k_+}}{E_{k_+} + E_{k_-}}. \tag{16}$$

This choice is argued by considering expression (10) for the corresponding zero-variance conditional PMF. The values $|\mathcal{X}_{G^{[i]}}|$ and $|\mathcal{X}_{G^{[-i]}}|$ are approximated by randomization and taking expectations. Note, similarly we approximate $f_1^*(1)$, as given in (9).

3.1 Complexity Analysis

We claim that the SIS algorithm with conditional PMF's computed according to (16) is FPRAS *for random graphs*. To justify our claim, we first discuss a related problem, counting the number of cliques in a given graph $G = G(V, E)$ with $|V| = n$. A subset of vertices $C \subset V$ is a *clique* if every two vertices in the subset are connected by an edge. Rasmussen [21] developed a randomized algorithm for counting cliques that is FPRAS for random graphs. Below we shall argue

(i). that the number of vertex covers and the number of cliques in a random graph are statistically the same;

(ii). and that our algorithm translates one-to-one to Rasmussen's algorithm.

These two arguments justify that our algorithm for counting vertex covers is FPRAS for random graphs.

Argument (i).

Let $\mathscr{G}$ be a random graph with n vertices and edges chosen according to the Edgar

Gilbert model. Consider an instance (or realisation) $G = G(V, E)$ of the random graph, and let $T \subset V$ be a vertex cover in G. Then we know that $S = V \setminus T$ is an independent set in G. And we see immediately that S is a clique in the complement graph $\overline{G} = G(V, \overline{E})$ (an edge $e \in \overline{E}$ iff $e \notin E$). Reversely, let $C \subset V$ be a clique in $\overline{G}$. Then, C is an independent set in G, and thus $V \setminus C$ a vertex cover in G. Hence, for every fixed graph G it holds that the number of vertex covers in G equals the number of cliques in $\overline{G}$:

$$|\mathcal{X}_G^{\mathrm{vc}}| = |\mathcal{X}_{\overline{G}}^{\mathrm{cl}}|.$$

Furthermore, in the randomization model every edge is chosen or not (to be present in the graph) with equal probability 1/2. That means that the random graph $\mathscr{G}$ and its complement $\overline{\mathscr{G}}$ have the same distribution, which we express by $\mathscr{G} \stackrel{\mathcal{D}}{=} \overline{\mathscr{G}}$. Consequently, this holds also for the associated number of vertex covers and cliques:

$$|\mathcal{X}_{\mathscr{G}}^{\mathrm{vc}}| = |\mathcal{X}_{\overline{\mathscr{G}}}^{\mathrm{cl}}| \stackrel{\mathcal{D}}{=} |\mathcal{X}_{\mathscr{G}}^{\mathrm{cl}}|.$$

Argument (ii).

First we describe the algorithm of Rasmussen [21] for counting cliques in a graph $G = G(V, E)$ with $|V| = n$. It is an importance sampling algorithm producing a binary n-tuple $\boldsymbol{x} \in \{0, 1\}^n$ with probability $g(\boldsymbol{x})$. Similarly as for vertex covers, the output of the algorithm is $Z^{\mathrm{cl}} = \mathbb{I}\{V(\boldsymbol{X}) \in \mathcal{X}_G^{\mathrm{cl}}\}/g(\boldsymbol{x})$ so that this estimator is unbiased. Furthermore, although the algorithm in [21] is presented as a recursive method, it is easy to see that it can be considered as being a sequential method like our SIS Algorithm 2.1 for vertex covers. In fact, it is exactly the same as our algorithm (obviously with probabilities $g_i(x_i|x_1, \ldots, x_{i-1})$ in stead of

$f_i(x_i|x_1,\ldots,x_{i-1})$), except for the rule with the *if* statement on lines 3-4. This should be replaced by checking that v_i cannot be added to the clique in which case $x_i = 0$. The entire algorithm is given in Appendix A.

Now, let us be more specific on the conditional probabilities $g_i(x_i|x_1,\ldots,x_{i-1})$ in Rasmussen's algorithm. For that purpose we introduce subgraphs $H^{[i]}$ and $H^{[-i]}$ $(i = 1,\ldots,n-1)$.

Definition 3.2 *Let be given binary $x_1,\ldots,x_{i-1}$ indicating whether nodes $v_j, j < i$, are part of a vertex subset $V(\boldsymbol{x})$.*

1. $i = 1$.

- $H^{[-1]} = G(V_1, E_1)$ *where* $V_1 = \{v_2,\ldots,v_n\}$, *and* $E_1 = \{(v_j,v_k)|v_j,v_k \in V_1\} \cap E$.
- $H^{[1]} = G(V_2, E_2)$ *where* $V_2 = \{v_k \in V_1|(v_1,v_k) \in E\}$, *and* $E_2 = \{(v_j,v_k)|v_j,v_k \in V_2\} \cap E$;

2. $i = 2,\ldots,n-1$.

- $H^{[-i]} = G(V_1, E_1)$ *where*

  $$V_1 = \{v_{i+1},\ldots,v_n\}\backslash\{v_k|k \geq i{+}1, \text{and } \exists\, j \leq i{-}1 \text{with } x_j = 1, (v_j,v_k) \notin E\},$$

  and $E_1 = \{(v_j,v_k)|v_j,v_k \in V_1\} \cap E$;
- $H^{[i]} = G(V_2, E_2)$ *where*

  $$V_2 = \{v_k \in V_1|(v_i,v_k) \in E\},$$

  and $E_2 = \{(v_j,v_k)|v_j,v_k \in V_2\} \cap E$.

The proposal PMF $g(\cdot)$ in the algorithm is obtained by specifying the conditional PMF's $g_i(1|x_1,\ldots,x_{i-1})$ for any binary tuple $(x_1,\ldots,x_{i-1})$. Let be given such a tuple. Suppose that node v_i may or may not be included in the clique (otherwise $x_i = 0$ with probability 1). Then we determine the associated subgraphs $H^{[-i]}$ and $H^{[i]}$ from Definition 3.2. Let these graphs have k_- and k_+ vertices, respectively. Then we let

$$g_i(1|x_1,\ldots,x_{i-1}) = \frac{E_{k_+}}{E_{k_+} + E_{k_-}}, \tag{17}$$

where E_k is defined in (14).

Finally we have to show the equivalence between the two algorithms (Algorithm 2.1 and A.1). First, denote by $\overline{x}_i = 1 - x_i$ and by $\overline{\boldsymbol{x}} = (\overline{x}_1,\ldots,\overline{x}_n)$, called the *complement binary n-tuple*. Let be given a graph $G = G(V,E)$ with $|V| = n$, and let $\overline{G}$ be its complement. For any binary n-tuple $\boldsymbol{x} \in \{0,1\}^n$ we have

$$V(\boldsymbol{x}) \in \mathcal{X}_G^{\mathrm{vc}} \quad \Leftrightarrow \quad V(\overline{\boldsymbol{x}}) \in \mathcal{X}_{\overline{G}}^{\mathrm{cl}}.$$

Now we claim the following.

(a). Apply our Algorithm 2.1 to G with the proposal $f(\cdot)$ given by (16), resulting in a binary n-tuple $\boldsymbol{x}$ with probability $f(\boldsymbol{x})$ for which holds that the associated vertex set $T = V(\boldsymbol{x})$ is a vertex cover; i.e., $T \in \mathcal{X}_G^{\mathrm{vc}}$. The vertex set $C = V(\overline{\boldsymbol{x}})$ of the complement tuple is a clique in the complement graph; i.e., $C \in \mathcal{X}_{\overline{G}}^{\mathrm{cl}}$. We shall show that $g(\overline{\boldsymbol{x}}) = f(\boldsymbol{x})$, meaning that when we would have applied Rasmussen's Algorithm A.1 to $\overline{G}$ with the proposal $g(\cdot)$ given by (17), we would have generated the clique C with the same probability as generating vertex cover T in graph G. Note that $C \cap T = \emptyset$ and $C \cup T = V$.

(b). Apply Rasmussen's Algorithm A.1 to G with the proposal $g(\cdot)$ given by (17),

resulting in a binary n-tuple $\boldsymbol{x}$ with probability $g(\boldsymbol{x})$ for which holds that the associated vertex set $C = V(\boldsymbol{x})$ is a clique; i.e., $C \in \mathcal{X}_G^{\mathrm{cl}}$. The vertex set $T = V(\overline{\boldsymbol{x}})$ of the complement tuple is a vertex cover in the complement graph; i.e., $T \in \mathcal{X}_{\overline{G}}^{\mathrm{vc}}$. We shall show that $f(\overline{\boldsymbol{x}}) = g(\boldsymbol{x})$, meaning that when we would have applied our Algorithm 2.1 to $\overline{G}$ with the proposal $f(\cdot)$ given by (16), we would have generated the vertex cover T with the same probability as generating clique C in graph G.

Claims (a) and (b) establish the one-to-one property of the two algorithms.

Proofs of claims (a) and (b).

In Appendix B.

4 Algorithm B

In this section we shall develop an improved proposal PMF for executing SIS for counting vertex covers. Denoting this proposal by $\tilde{f}(\cdot)$ we mean that it improves the proposal $f(\cdot)$ because the the associated importance sampling estimator has lower variance:

$$\mathbb{V}\mathrm{ar}[Z_{\tilde{f}}] \leq \mathbb{V}\mathrm{ar}[Z_f].$$

The idea is the same as getting expression (16) by approximating the values in the expression of the zero-variance conditional probabilities by randomization and taking expectations. First we shall introduce the new probability model for random graphs.

4.1 Vertex Cover Relaxation

In this section we introduce the vertex cover relaxation method. This is a randomization technique for which the expected number of vertex covers can be computed efficiently.

Recall that we consider a given graph $G = G(V, E)$, with $|V| = n$, and that we have labeled the vertices $v_1, \ldots, v_n$ in some order. We keep the order fixed, and thus we can define *the set of downstream neigbours* of vertex v_i:

$$D_i = \{v_j \in V | j \geq i + 1, \text{ and } (v_i, v_j) \in E\}, \quad i = 1, \ldots, n - 1.$$

Let $d_i = |D_i|$ be the cardinality of this set. Note that $d_i \leq n - i$ because the graph is simple. Therefore we can define a vector of probabilities as follows.

Definition 4.1 *The vector of downstream probabilities is* $\boldsymbol{p} = (p_1, \ldots, p_n)$*, where*

$$p_i = \frac{d_i}{n-1}, \quad i = 1, \ldots, n-1; \quad p_n = 0.$$

Note that $\boldsymbol{p}$ is not a probability vector!

Consider now a probability space Ω_G of all graphs $G' = G(V, E')$ with the same vertex set V, but where each possible edge $(v_i, v_j),\ j > i$ is present in E' with probability p_i. Note that

$$\begin{aligned} p_i = 0 \quad &\Rightarrow \quad (v_i, v_j) \notin E' \ \ \forall j > i, \\ p_i = 1 \quad &\Rightarrow \quad (v_i, v_j) \in E' \ \ \forall j > i. \end{aligned}$$

Example 4.1 *Two simple examples are the bridge graph and the star graph. They are depicted in Figure 4.*

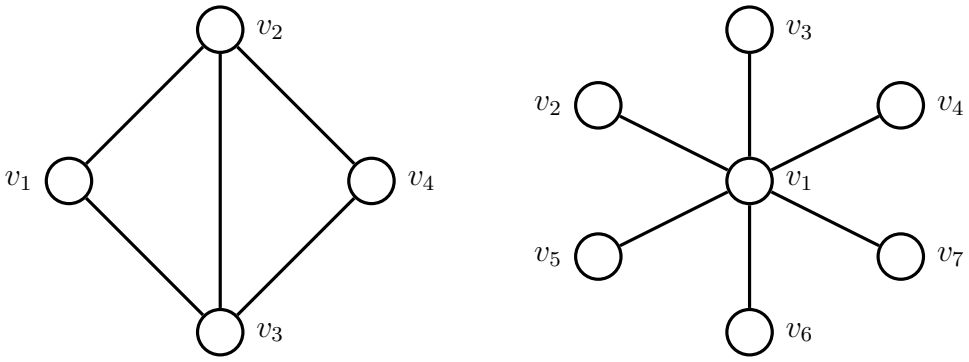

Figure 4: Left panel: bridge graph. Right panel: star graph.

Concerning the bridge graph, one can easily observe that v_1 *is connected to* v_2 *and* v_3*,* v_2 *is connected to* v_3 *and* v_4 *and, finally,* v_3 *is connected only to* v_4*, so that we have* $d_1 = 2$*,* $d_2 = 2$*,* $d_3 = 1$*, respectively. The last vertex* v_4 *has zero connections under our relaxation, so* $d_4 = 0$ *and* $\boldsymbol{p} = (\frac{2}{3}, \frac{2}{2}, \frac{1}{1}, 0)$*. For the star graph we get* $\boldsymbol{p} = (1, 0, 0, 0, 0, 0, 0)$. □

In what follows it will be convenient to consider also the *vector of complementary downstream probabilities* $\boldsymbol{q} = (q_1, \cdots, q_n)$, where

$$q_i = 1 - p_i = 1 - \frac{d_i}{n-1}, \quad i = 1, \ldots, n-1; \quad q_n = 1. \tag{18}$$

In the probability model, q_i is the probability that the edge (v_i, v_j) $(j > i)$ is not present in G'. Given a graph $G = G(V, E)$, the calculation of vector $\boldsymbol{q}$ is straightforward.

Denote by $\mathscr{G}$ a random graph in Ω_G, with probability law $\mathbb{P}_G$ given above, and with associated expectation $\mathbb{E}_G$. Clearly, for any realization $G' \in \Omega_G$

$$\mathbb{P}_G(\mathscr{G} = G') = \prod_{i=1}^{n-1} p_i^{d_i'} q_i^{n-i-d_i'},$$

where $0^0 = 1$ and d_i' is defined in the same way as d_i; that is, the cardinality of the set of downstream neigbors in the graph G'.

4.2 Expected Number of Relaxed Vertex Covers

Let $\mathcal{X}_G$ be the set of all vertex covers of the given graph $G = G(V, E)$ and $|\mathcal{X}_G|$ be its cardinality. Similarly, for any graph $G' \in \Omega_G$ the set of vertex covers is $\mathcal{X}_{G'}$. In this section we are interested in the expected number of vertex covers of the random graph $\mathscr{G}$:

$$\mathbb{E}_G|\mathcal{X}_{\mathscr{G}}| = \sum_{G' \in \Omega_G} \mathbb{P}_G(\mathscr{G} = G')|\mathcal{X}_{G'}| \ .$$

Example 4.2 *Consider the bridge graph given in Example 4.1 with the vector of probabilities* $\boldsymbol{p} = (\frac{2}{3}, \frac{2}{2}, \frac{1}{1}, 0)$. *The set of 8 possible graphs in the probability space* Ω_G *is summarized in Figure 5. Note that* $p_2 = 1$, *hence each graph must contain both edges* (v_2, v_3) *and* (v_2, v_4). *Similarly, because* $p_3 = 1$, *edge* (v_3, v_4) *is always present.*

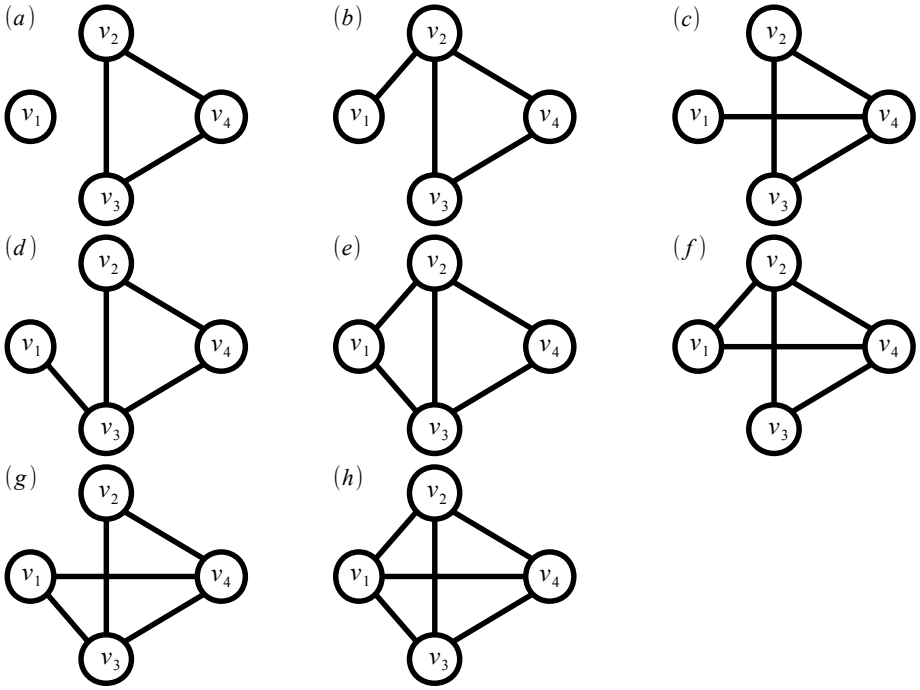

Figure 5: Sample space associated with the bridge graph.

Graph (a) *is generated with probability* $(\frac{1}{3})^3$*; graphs* $(b), (c), (d)$ *with probability* $\frac{2}{3}(\frac{1}{3})^2$*; graphs* $(e), (f), (g)$ *with probability* $\frac{1}{3}(\frac{2}{3})^2$*, and graph* (h) *with probability* $(\frac{2}{3})^3$*. The corresponding number of vertex covers for graphs* $(a), (b), \ldots, (h)$ *is* $8, 7, 7, 7, 6, 6, 6, 5$*. For instance, consider graph* (a)*, and its subgraph of nodes* v_2, v_3, v_4 *(with their incident edges). The vertex covers of this subgraph are* $\{v_2, v_3\}$, $\{v_2, v_4\}$, $\{v_3, v_4\}$, $\{v_2, v_3, v_4\}$*. Because node* v_1 *has no incident edge, these four sets are also covers for the whole graph. Of course, with node* v_1 *added, these sets remain vertex covers giving a total of eight covers. Hence, we can compute the expected number of vertex covers*

$$\mathbb{E}_G|\mathcal{X}_{\mathcal{G}}| = \left(\frac{1}{3}\right)^3 8 + 3\frac{2}{3}\left(\frac{1}{3}\right)^2 7 + 3\frac{1}{3}\left(\frac{2}{3}\right)^2 6 + \left(\frac{2}{3}\right)^3 5 = 6\ .$$

□

A crucial property of the proposed relaxation is summarized next.

Proposition 4.1 *There exists a deterministic polynomial time algorithm that computes* $\mathbb{E}_G|\mathcal{X}_{\mathscr{G}}|$ *analytically.*

To prove this proposition we first establish some auxiliary results. Suppose that a subset of vertices $S \subset V$ has size k. Then we denote the ordered vertices of S by $v_{i_1}, v_{i_2}, \ldots, v_{i_k}$; that is, $i_j < i_{j+1}$ for all $j = 1, 2, \ldots, k-1$.

Lemma 4.1 *Given the vector of complementary downstream probabilities* $\boldsymbol{q}$ *of* (18)*, we have*

$$\mathbb{E}_G|\mathcal{X}_{\mathscr{G}}| = \sum_{k=0}^{n} \sum_{\substack{S \subset V \\ |S|=k}} \prod_{j=1}^{k-1} q_{i_j}^{k-i_j}.$$

Proof. Define, for $k = 0, \ldots, n$, $A(n,k)$ to be the expected number of vertex covers of size $n-k$ in the random graph $\mathscr{G}$. Again we use the property that, when $T \subset V$ forms a vertex cover of a graph, its complement $S = V \setminus T$ forms an independent set. Thus,

$$\begin{aligned} A(n,k) &= \sum_{\substack{T \subset V \\ |T|=n-k}} \mathbb{P}_G(T \text{ forms a vertex cover}) \\ &= \sum_{\substack{S \subset V \\ |S|=k}} \mathbb{P}_G(S \text{ forms an independent set}) \end{aligned}$$

Let $S = \{v_{i_1}, \ldots, v_{i_k}\}$. Then, S is an independent set if and only if none of the edges (v_{i_j}, v_{i_ℓ}) is chosen for all $j = 1, \ldots, k-1$ and all $\ell = j+1, \ldots, k$. This happens exactly with probability $\prod_{j=1}^{k-1} q_{i_j}^{k-i_j}$. The proof is complete by noting

that

$$\mathbb{E}_G|\mathcal{X}_{\mathscr{G}}| = \sum_{k=0}^{n} A(n,k). \tag{19}$$

□

If, for example, the vector of probabilities $\boldsymbol{q}$ satisfies $q_i \equiv q \in (0,1)$ for all i, we obtain

$$A(n,k) = \binom{n}{k} q^{\binom{k}{2}} .$$

Since the last simplification is not valid for general graphs, we next explain how to calculate $A(n,k)$ analytically using a dynamic programming type of recursion.

Consider, for $m = 1, 2, \ldots, n$, the subgraph $G_m = G(V_m, E_m)$ of G consisting of the vertices $V_m = \{v_{n-m+1}, \ldots, v_n\}$ and their incident edges $E_m \subset E$. Similarly, we define the random graph $\mathscr{G}_m$ having V_m as vertex set, and edges chosen randomly according to the vector of probabilities $\boldsymbol{p}$ (of the original graph G). Finally, we define $A(m,k)$ to be the expected number of vertex covers of size $m-k$, $k \le m$, in the random graph $\mathscr{G}_m$ for $m = 1, \ldots, n$. The idea is to compute the $A(m,k)$ numbers via a recursion by considering iteratively vertex covers in $V_1, V_2, \ldots$. Furthermore, the vertex covers of size $m+1-k$ in V_{m+1} can be decomposed into vertex covers of size $m+1-k$ in V_m, and in vertex covers of size $m-k$ in V_m. The precise recursion is formulated in the following.

Lemma 4.2 *For $m = 1, 2, \ldots, n-1$ and $k = 1, \ldots, m+1$, we have the recursion*

$$A(m+1,k) = q_{n-m}^{k-1}\, A(m,k-1) + A(m,k) \;, \tag{20}$$

where $A(m,0) = 1$ for $m = 1, \ldots, n$, and $A(1,1) = 1$.

Proof. Again we will use the property that, when $T \subset V_m$ forms a vertex cover of a graph, its complement $S = V_m \setminus T$ forms an independent set. Let S be an independent set, and $\#\{U\}$ stand for the size of set U.

$$\begin{aligned} A(m+1,k) &= \mathbb{E}_G[\#\{S \text{ of size } k \text{ in } \{v_{n-m}, v_{n-m+1}, \ldots, v_n\}\}] \\ &= \mathbb{E}_G[\#\{S \text{ of size } k \text{ in } \{v_{n-m}, \ldots, v_n\}\} \text{ and } v_{n-m} \in S] \\ &\quad + \mathbb{E}[\#\{S \text{ of size } k \text{ in } \{v_{n-m}, \ldots, v_n\}\} \text{ and } v_{n-m} \notin S] \end{aligned}$$

The two terms of the decomposition are computed as follows.

- First term: the remaining nodes $S \setminus \{v_{n-m}\}$ form an independent set of size $k-1$ in $\{v_{n-m+1}, \ldots, v_n\}$, and none of these $k-1$ nodes is connected with v_{n-m}. Since $A(m, k-1)$ is the expected number of such independent sets, and since choosing edges between nodes is independent of anything else, the first term yields $q_{n-m}^{k-1} A(m, k-1)$.

- Second term: the remaining nodes $S \setminus \{v_{n-m}\}$ form an independent set of size k in $\{v_{n-m+1}, \ldots, v_n\}$, thus it does not matter whether any of these nodes is connected with v_{n-m} or not. Hence, the second term yields $A(m, k)$.

□

The algorithm for calculating $\mathbb{E}_G|\mathcal{X}_{\mathcal{G}}|$ for a given graph $G = G(V, E), |V| = n$, can be summarized as follows.

Algorithm 4.1 ***Calculating Number of Relaxed Covers***

Input: $G = G(V, E)$

Output: $\mathbb{E}_G|\mathcal{X}_{\mathcal{G}}|$

1: $\boldsymbol{q} \leftarrow$ *calculate the vector of probabilities as in* (18)*;*

2: $\forall k \in \{0, \cdots, n\}$ *calculate* $A(n,k)$ *using recursion* (20)*;*

3: ***return*** $\mathbb{E}_G|\mathcal{X}_{\mathscr{G}}|$ *as in* (19)*.*

Proof of Proposition 4.1. Step (1) of the algorithm takes $\mathcal{O}(n)$ time (see Remark 4.1); step (2) can be completed in $\mathcal{O}(n^2)$, and step (3) takes only linear time $\mathcal{O}(n)$. Since all those steps are governed by (2) conclude that the overall complexity of Algorithm 4.1 is $\mathcal{O}(n^2)$ and Proposition 4.1 follows. □

Remark 4.1 *In order to get an efficient implementation of the above algorithm, the graph should use the following data structure.*

$$
\begin{aligned}
v_1 &\to \{v_{a_1^1}, \ldots, v_{a_{k_1}^1}\} \\
&\vdots \\
v_i &\to \{v_{a_1^i}, \ldots, v_{a_{k_i}^i}\} \\
&\vdots \\
v_n &\to \emptyset
\end{aligned}
$$

Each entry v_i *shell have a list of adjacent vertices* $\{v_{a_1^i}, \ldots, v_{a_{k_i}^i}\}$ *such that each adjacent vertex index* $a_1^i, \ldots, a_{k_i}^i$ *is greater than* i*. Having in mind the above structure, step (1) of the algorithm can be calculated in* $\mathcal{O}(n)$ *time. This data structure is also very helpful during the execution of the main Sequential Importance Sampling procedure due to the fact that a vertex removal can be implemented in* $\mathcal{O}(n\log(n))$ *time provided that the adjacent indexes are ordered.*

Algorithm 4.1 can also count vertex covers exactly in some cases. As an example consider the star graph on the right panel of Figure 4, with n nodes. It is not

difficult to determine the exact number of vertex covers in this case. If the central vertex participates in the cover, then there are 2^{n-1} covers, because any combination of the remaining $n-1$ vertices yields a valid cover. If the central vertex it is not in the cover, then all the remaining vertices must be part of one cover and we conclude that the exact number of vertex covers in the star graph is $2^{n-1}+1$. If we take the ordering of nodes such that v_1 is the central vertex, then the induced vector of probabilities will be $\boldsymbol{p} = (1, 0, \ldots, 0)$. Now, running Algorithm 4.1 with a star graph as an input will result in $2^{n-1}+1$. In general, we have the following result.

Proposition 4.2 *Given that an instance $G = G(V, E)$ induces a vector of downstream probabilities $\boldsymbol{p} = (p_1, \ldots, p_n)$ where each $p_i \in \{0, 1\}$, Algorithm 4.1 provides the exact number of vertex covers, that is, $\mathbb{E}_G|\mathcal{X}_{\mathscr{G}}| = |\mathcal{X}_G|$.*

Proof. One way to proceed is by induction on the number of vertex covers combined with equation (20). However, it is much simpler to notice that given a vector of probabilities $\boldsymbol{p} = (p_1, \ldots, p_n)$ where each $p_i \in \{0, 1\}$, there is only one graph G with $\boldsymbol{p}$ as its vector of downstream probabilities. This observation follows easily from the construction process of the random graph using this particular $\boldsymbol{p}$. For $|\Omega_G| = 1$ we obtain $\mathbb{E}_G|\mathcal{X}_{\mathscr{G}}| = |\mathcal{X}_G|$. □

It follows from Proposition 4.2 and from Algorithm 4.1 that if there is an ordering in $G = G(V, E)$ such that the vector of complementary downstream probabilities $\boldsymbol{q}$ satisfies $q_i \in \{0, 1\}$, then the number of vertex covers is available analytically and can be calculated in $\mathcal{O}(|V|^2)$ time.

4.3 The Proposal PMF for SIS

Finally, we return to the common issue in our study of importance sampling for counting vertex covers in a graph. We specify the conditional probabilities $f_i(x_1, \ldots, x_{i-1})$ of the proposal PMF for any binary tuple $(x_1, \ldots, x_{i-1})$. Let be given such a tuple. Suppose that node v_i may or may not be included in the vertex cover (otherwise $x_i = 1$ with probability 1). Then we determine the associated subgraphs $G^{[i]}$ and $G^{[-i]}$ from Definition 2.1, and we let

$$f_i(1|x_1, \ldots, x_{i-1}) = \frac{\mathbb{E}_G|\mathcal{X}_{\mathscr{G}^{[i]}}|}{\mathbb{E}_G|\mathcal{X}_{\mathscr{G}^{[-i]}}| + \mathbb{E}_G|\mathcal{X}_{\mathscr{G}^{[i]}}|} , \tag{21}$$

Thus, in essence, we approximate the values $|\mathcal{X}_{G^{[i]}}|$ and $|\mathcal{X}_{G^{[-i]}}|$ in the expression of the zero-variance probability (10) again by randomization and taking expectations. The difference with the proposal (16) in Section 3 is the implementeation of the probability model. Note that both expectations in(21) are readily computed using Algorithm 4.1.

The performance of Algorithm B will be studied in the next section.

5 Numerical Results

Summarizing, we introduced two algorithms, A and B; both apply the SIS algorithm 2.1. Algorithm A uses proposal PMF with conditional probabilities given in (16); whereas in Algorithm B these probabilities are given in (21).

In this section we consider the performances of Algorithms A and B on different models. We performed all computation on Core i5 laptop with 4GB RAM. The reported CPU time is measured in seconds. For smaller problems we are able to compute the exact count. In that case we report the numerical relative

error of the estimates; in the other cases we report the statistical relative error estimated according to (4).

To our knowledge, no randomized algorithms have been developed dedicated to the vertex cover counting problem. Thus to compare our algorithm we have considered the following more generally applicable methods which have shown to produce good performance.

- *Cachet* is exact model counting software introduced by Sang et al. in [26]. This method uses the well known SAT solver Chaff [20] and combines component caching with traditional clause learning within the setup of model counting.

- *SampleSearch* is a probabilistic model counting technique proposed by Gogate and Dechter in [7, 10]. The method can deliver upper and lower bounds on counting problems and is based on sampling from the search space of a Boolean formula. Similarly to *Cachet*, it also uses a DPLL-based SAT solvers during the execution in order to construct the sampling search space.

5.1 Random Graphs

In order to test the FPRAS property of Algorithm A that was discussed in Section 3.1 we performed the following experiment. For each $n = 5, 10, 15, \ldots, 100$, we generated 40 random graphs from the Edgar Gilbert model. Next, we ran Algorithm B with $N = 1000$ on each graph. Also we ran Algorithm B with sample size of $N = 100$. Our conjecture that the proposal (21) used in Algorithm B is more delicate in the sense that it is closer to the zero variance PMF, is verified numerically. For each n, we report the average cv and the average *Relative Time Variance* (RTV). The latter is used to compare different algorithms; it is defined

as the simulation time in seconds multiplied by the squared relative error. Figures 6 and 7 summarize the obtained results. One can clearly observe that even with this favorable settings, for which Algorithm A is FPRAS, the proposal (21) has clear superiority.

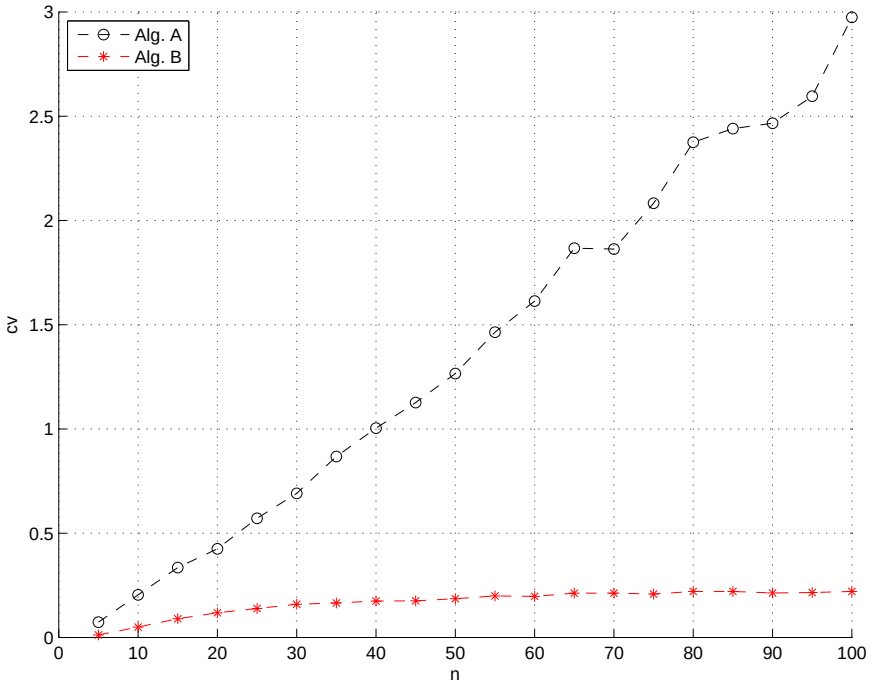

Figure 6: cv as a function of n.

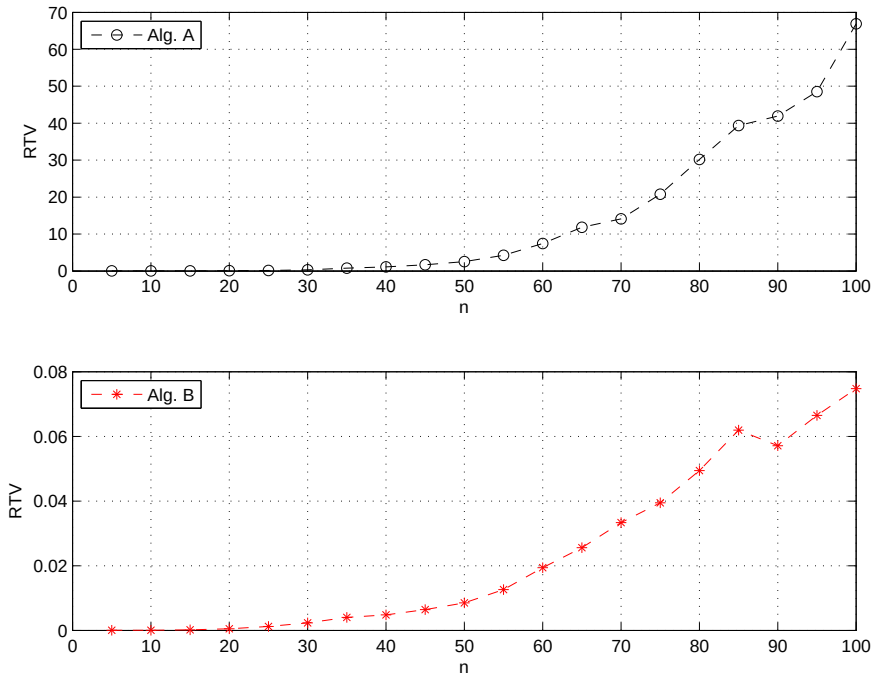

Figure 7: RTV as a function of n.

5.2 Model 1

A graph with $|V| = 100$. The edge set E was constructed according to an adapted Edgar Gilbert random graph model, as follows. For each edge (v_i, v_j) from all $\binom{|V|}{2}$ possible edges we first generated a random number from the uniform (0,1) distribution, say p; then the edge was included in the graph with probability p, namely by generating u from the uniform (0,1) distribution and checking $u < p$. All random drawings were executed independently The instance that we obtained in this way contained $|E| = 2,432$ edges. Then we executed our Algorithm B with sample size $N = 100$. We report the computed estimate $\widehat{|\mathcal{X}_G|}$, the required computing time CPU, standard error, and the numerical relative error because the *cachet* method delivered the exact count. In order to check stability and robustness of the algorithm, we repeated ten times the estimation procedure with the same graph. The following table summarizes the results.

Table 1: Performance of 10 runs of Algorithm B on Model 1 with sample size $N = 100$.

Run	$\widehat{\lvert\mathcal{X}_G\rvert}$	$\widehat{\mathrm{RE}}$	error	CPU
1	2.481×10^5	1.29×10^{-2}	1.62×10^{-2}	2.065
2	2.437×10^5	4.87×10^{-3}	1.73×10^{-2}	2.213
3	2.431×10^5	7.38×10^{-3}	1.62×10^{-2}	2.054
4	2.419×10^5	1.24×10^{-2}	1.65×10^{-2}	1.984
5	2.468×10^5	7.55×10^{-3}	1.49×10^{-2}	1.878
6	2.463×10^5	5.49×10^{-3}	1.71×10^{-2}	1.826
7	2.412×10^5	1.52×10^{-2}	1.81×10^{-2}	1.995
8	2.517×10^5	2.75×10^{-2}	1.31×10^{-2}	2.143
9	2.443×10^5	2.54×10^{-3}	1.53×10^{-2}	1.935
10	2.372×10^5	3.18×10^{-2}	1.55×10^{-2}	2.212
Average	2.444×10^5	1.28×10^{-2}	1.60×10^{-2}	2.031

For comparison:

- Running *cachet* delivers an exact solution of 244941 in 0.75 seconds.
- Running *SampleSearch* 10 times provides an average of 196277 in 60 seconds with estimated relative error of about 20%.

For this example *cachet* provides the exact solution in the fastest time. *SampleSearch* provides a good lower bound, but at a high CPU time cost, and Algorithm B performs as second best.

5.3 Model 2

A graph with $|V| = 300$ and $|E| = 21,094$, generated in the same manner as Model 1. The SIS performance is given in the following table.

Table 2: Performance of 10 runs of Algorithm B for Model 2 with $N = 100$.

Run	$\widehat{\lvert\mathcal{X}_G\rvert}$	$\widehat{\mathrm{RE}}$	error	CPU
1	1.360×10^{14}	4.29×10^{-2}	4.06×10^{-2}	69.33
2	1.298×10^{14}	4.64×10^{-2}	6.98×10^{-3}	78.36
3	1.308×10^{14}	4.22×10^{-2}	1.33×10^{-3}	102.46
4	1.277×10^{14}	4.23×10^{-2}	2.23×10^{-2}	98.04
5	1.327×10^{14}	3.76×10^{-2}	1.56×10^{-2}	103.36
6	1.241×10^{14}	3.86×10^{-2}	5.01×10^{-2}	112.55
7	1.270×10^{14}	3.92×10^{-2}	2.78×10^{-2}	117.71
8	1.293×10^{14}	4.40×10^{-2}	1.06×10^{-2}	84.02
9	1.258×10^{14}	3.83×10^{-2}	3.72×10^{-2}	97.73
10	1.279×10^{14}	3.97×10^{-2}	2.16×10^{-2}	91.81
Average	1.291×10^{14}	4.11×10^{-2}	2.34×10^{-2}	95.54

For comparison:

- Running *cachet* delivers an exact solution of 1.306×10^{14} in about 17 minutes.
- Running *SampleSearch* ten times provides an average of 5.791×10^{13} in $1,200$ seconds with estimated relative error of about 55%.

5.4 Model 3

A graph with $|V| = 1,000$. In this model, edges were again randomly included one by one according to $u < p$, but now $p \in (0,1)$ was generated from a truncated normal distribution on the interval $[0,1]$ with $\mu = 0.1$ and $\sigma = 0.1$; as before, u came from the uniform (0,1) distribution. We obtained an instance with $|E| = 64,251$. The instance size is too large for an exact solution, therefore we give the (estimated) relative error of the estimator, denoted $\widehat{\mathrm{RE}}$, and computed as in (4). The results are summarized bellow.

Table 3: Performance of 10 runs of Algorithm B on Model 3 with $N = 100$.

Run N_0	$\widehat{\lvert\mathcal{X}_G\rvert}$	$\widehat{\mathrm{RE}}$	CPU
1	4.384×10^{32}	4.450×10^{-2}	661.5
2	4.058×10^{32}	4.151×10^{-2}	703.1
3	4.014×10^{32}	4.814×10^{-2}	691.9
4	4.137×10^{32}	4.215×10^{-2}	721.3
5	4.220×10^{32}	4.437×10^{-2}	691.0
6	4.124×10^{32}	4.555×10^{-2}	688.6
7	4.422×10^{32}	4.788×10^{-2}	667.0
8	4.184×10^{32}	4.563×10^{-2}	682.0
9	4.234×10^{32}	4.103×10^{-2}	698.9
10	4.261×10^{32}	4.813×10^{-2}	648.6
Average	4.204×10^{32}	4.489×10^{-2}	685.4

For comparison:

- *cachet* was unable to deliver a solution within 2 days of CPU time. The lower bound of 3.439×10^9 was only supplied.
- *SampleSearch* failed to initialize possibly due to the large size of the problem.

5.5 Model 4

A graph with $|V| = 1,000$ and $|E| = 249,870$. This time p was generated from a truncated Normal distribution with $\mu = 0.5$ and $\sigma = 0.3$. The results are summarized bellow.

Table 4: 10 runs of Algorithm B Model 4 with $N = 100$.

Run N_0	$\widehat{\lvert\mathcal{X}_G\rvert}$	$\widehat{\mathrm{RE}}$	CPU
1	2.749×10^{11}	1.608×10^{-2}	1841
2	2.762×10^{11}	1.531×10^{-2}	1847
3	2.848×10^{11}	1.720×10^{-2}	1658
4	2.737×10^{11}	1.274×10^{-2}	1562
5	2.795×10^{11}	1.521×10^{-2}	1642
6	2.819×10^{11}	1.591×10^{-2}	1853
7	2.764×10^{11}	1.701×10^{-2}	1756
8	2.776×10^{11}	1.768×10^{-2}	1544
9	2.698×10^{11}	1.468×10^{-2}	1767
10	2.778×10^{11}	1.605×10^{-2}	1708
Average	2.773×10^{11}	1.579×10^{-2}	1718

For comparison:

- *cachet* was timed out after 2 days and was unable to deliver a solution. The lower bound of 9.601×10^{10} was supplied.
- *SampleSearch* failed to initialize. We speculate that the reason for this is that the problem is too large.

5.6 Nonrandom graphs

Finally, consider nonrandom graphs, namely the hypercube graphs H_n, $n = 4, 5, 6, 7$ with 2^n vertices and $n2^{n-1}$ edges, see [12]. In graph theory, 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. For nonrandom graphs, we expect our algorithm's performance to deteriorate. While dealing with random graphs, the distribution of adjacent vertices of some given vertex is some that "uniform" so hopefully our dynamic

programming will provide a good approximation because it is also based on random model. For nonuniform graphs, we can not hope to enjoy the same behavior.

The exact number of vertex covers for H_4, H_5 and H_6 is know to be 743, 254475 and 1.976×10^{10}, respectively. The following table summarizes the empirical performance, where we set the sample size N to be $50, 250, 1500$ and 10^4 for H_4, H_5, H_6 and H_7, respectively, so that the estimated relative error was kept below 3%.

Table 5: Performance of the Algorithm B for the Hypercube graphs.

Instance	$\widehat{\lvert\mathcal{X}_G\rvert}$	$\widehat{\mathrm{RE}}$	CPU
H_4	745.9	2.87×10^{-2}	0.008
H_5	2.550×10^5	2.86×10^{-2}	0.157
H_6	1.983×10^{10}	2.67×10^{-2}	4.841
H_7	7.819×10^{19}	2.89×10^{-2}	199.8

6 Concluding Remarks

In this article we studied the problem of counting the number of vertex covers in a simple graph. This problem has #P-complete complixity classification as a function of the number vertices. We considered the application of sequential importance sampling (SIS) as a randomized approximation method to produce estimates of the counting number. The importance sampling simulation is executed by a proposal probability mass function (PMF) on the vertex covers. The crucial result was that we determined the expressions of the conditional densities of the zero-variance proposal PMF. These expressions refer to the number of vertex covers in specific subgraphs. Although this PMF is not implementable, by constructing a probability model on random graphs we replaced the expressions

in the zero-variance proposal PMF by corresponding expectations of the number of vertex covers of these random graphs. Moreover, these expectations were easily computed.

In fact, we introduced two of these probability models and analysed the resulting SIS algorithms. The first probability model is based on a classic model for random graphs, resulting in Algorithm A. We proved that this algorithm shows polynomial complexity (FPRAS) for random graphs. The analysis of this result was based on a similar result for counting cliques in simple graphs. Algorithm B resulted from the second randomization model that used a probabilistic relaxation of vertex covers. By numerical experiments we compared the two Algorithms in terms of statistical performances of the associated estimators. Our findings were that Algorithm B outperforms Algorithm A. Algorithm B is easy to implement and the numerical results suggest that the practical performance is comparable with and sometimes significantly better than currently existing methods. For example, with a sample size as little as 100, we observed low relative errors on problems with large dimensionality, for which alternative methods fail.

Of interest as future research is to theoretically investigate how closely the proposal distribution described in this article approximates the zero-variance measure. In addition, of interest will be the development of similar relaxation techniques to other important graph counting problems.

ACKNOWLEDGMENT

We are thoroughly grateful to the anonymous reviewers for their valuable and constructive remarks and suggestions.

References

[1] Soren Asmussen and Peter W. Glynn. *Stochastic Simulation: Algorithms and Analysis*. Applications of Mathematics. Springer Science and Business Media, LLC, 2007.

[2] Joseph Blitzstein and Persi Diaconis. A sequential importance sampling algorithm for generating random graphs with prescribed degrees. *Internet Mathematics*, 6:487–520, 2010.

[3] Bella Bollobás. *Random Graphs*. Cambridge Studies in Advanced Mathematics. Cambridge University Press, 2001.

[4] Zdravko Botev and Dirk Kroese. Efficient Monte Carlo simulation via the generalized splitting method. *Statistics and Computing*, 22:1–16, 2012.

[5] Yuguo Chen, Persi Diaconis, Susan P. Holmes, and Jun S. Liu. Sequential Monte Carlo methods for statistical analysis of tables. *Journal of the American Statistical Association*, 100:109–120, March 2005.

[6] Mary Cryan and Martin Dyer. A polynomial-time algorithm to approximately count contingency tables when the number of rows is constant. *Journal of Computer and System Sciences*, 67:291–310, 2003.

[7] Rina Dechter and Vibhav Gogate. A new algorithm for sampling CSP solutions uniformly at random. In *Principles and Practice of Constraint Programming*, May 2006.

[8] Martin Dyer. Approximate counting by dynamic programming. In *Proceedings of the 35th ACM Symposium on Theory of Computing*, pages 693–699, 2003.

[9] Martin Dyer, Alan Frieze, and Mark Jerrum. On counting independent sets in sparse graphs. In *In 40th Annual Symposium on Foundations of Computer Science*, pages 210–217, 1999.

[10] Vibhav Gogate and Rina Dechter. Approximate counting by sampling the backtrack-free search space. In *Proceedings of the 22nd national conference on Artificial Intelligence - Volume 1*, AAAI'07, pages 198–203. AAAI Press, 2007.

[11] Carla P. Gomes, Jörg Hoffmann, Ashish Sabharwal, and Bart Selman. From sampling to model counting. In *IJCAI*, pages 2293–2299, 2007.

[12] Frank Harary, John P. Hayes, and Horng-Jyh Wu. A survey of the theory of hypercube graphs. *Computers & Mathematics with Applications*, 15(4):277 – 289, 1988.

[13] Mark Jerrum and Alistair Sinclair. The Markov chain Monte Carlo method: An approach to approximate counting and integration. In D. Hochbaum, editor, *Approximation Algorithms for NP-hard Problems*, pages 482–520. PWS Publishing, 1996.

[14] Mark Jerrum, Alistair Sinclair, and Eric Vigoda. A polynomial-time approximation algorithm for the permanent of a matrix with non-negative entries. *Journal of the ACM*, pages 671–697, 2004.

[15] Mark Jerrum, Leslie G. Valiant, and Vijay V. Vazirani. Random generation of combinatorial structures from a uniform distribution. *Theor. Comput. Sci.*, 43:169–188, 1986.

[16] Richard M. Karp. Reducibility among combinatorial problems. In R.E. Miller and J.W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum, New York, 1972.

[17] Richard M. Karp and Michael Luby. Monte-Carlo algorithms for enumeration and reliability problems. In *Proceedings of the 24th Annual Symposium on Foundations of Computer Science*, SFCS '83, pages 56–64, Washington, DC, USA, 1983. IEEE Computer Society.

[18] Richard M. Karp, Michael Luby, and Neal Madras. Monte-Carlo approximation algorithms for enumeration problems. *Journal of Algorithms*, 10:429–448, 1989.

[19] Jingcheng Liu and Pinyan Lu. FPTAS for counting monotone CNF. *CoRR*, abs/1311.3728, 2013.

[20] Matthew W. Moskewicz, Conor F. Madigan, Ying Zhao, Lintao Zhang, and Sharad Malik. Chaff: Engineering an efficient SAT solver. In *Annual ACM IEEE Design Automation Conference*, pages 530–535. ACM, 2001.

[21] Lars E. Rasmussen. Approximately counting cliques. *Random Struct. Algorithms*, 11(4):395–411, 1997.

[22] Reuven Rubinstein. The Gibbs cloner for combinatorial optimization, counting and sampling. *Methodology and Computing in Applied Probability*, 11:491–549, 2009.

[23] Reuven Rubinstein. Stochastic enumeration method for counting NP-hard problems. *Methodology and Computing in Applied Probability*, 15(2):249–291, 2013.

[24] Reuven Rubinstein, Andrey Dolgin, and Radislav Vaisman. The splitting method for decision making. *Communications in Statistics - Simulation and Computation*, 41(6):905–921, 2012.

[25] Reuven Rubinstein and Dirk Kroese. *Simulation and the Monte Carlo Method, Second Edition*. John Wiley and Sons, New York, 2007.

[26] Tian Sang, Fahiem Bacchus, Paul Beame, Henry Kautz, and Toniann Pitassi. Combining component caching and clause learning for effective model counting. In *Seventh International Conference on Theory and Applications of Satisfiability Testing*, 2004.

[27] Salil P. Vadhan. The complexity of counting in sparse, regular, and planar graphs. *SIAM Journal on Computing*, 31:398–427, 1997.

[28] Leslie G. Valiant. The complexity of enumeration and reliability problems. *SIAM Journal on Computing*, 8(3):410–421, 1979.

Appendix A Rasmusen Algorithm for counting the number of cliques

Algorithm A.1 ***Sequential Sampling of Cliques***

Input: Graph $G = G(V, E)$ *and proposal PMF* $g(\cdot)$.

Output: Importance weight of the generated clique.

1: $Z \leftarrow 1$, $\boldsymbol{x} \leftarrow (0, \ldots, 0)$, *so that* $C(\boldsymbol{x}) = \emptyset$.

2: ***for*** $i = 1 \rightarrow n$ ***do*** ▷ *Note that at stage* i, $C(\boldsymbol{x}) \subseteq \{v_1, \ldots, v_{i-1}\}$ *is a clique,*

3: ***if*** v_i *can not be added to the clique* ***then***

4: $x_i \leftarrow 0$

5: ***else***

6: $U \sim \mathsf{U}(0, 1)$

7: ***if*** $U \leq g(1|x_1, \ldots, x_{i-1})$ ***then***

8: $x_i \leftarrow 1$, $Z \leftarrow Z \times \frac{1}{g(1|x_1,\ldots,x_{i-1})}$

9: ***else***

10: $Z \leftarrow Z \times \frac{1}{g(0|x_1,\ldots,x_{i-1})}$

11: ***end if***

12: ***end if***

13: ***end for***

14: $Z \leftarrow Z \times \mathbb{I}\{C(\boldsymbol{x}) \in \mathcal{X}_G^{\mathrm{cl}}\}$

15: ***return*** *Repeat the above procedure* N *times to generate* $Z_1, \ldots, Z_N$, *and deliver the average* $\frac{1}{N}\sum_{i=1}^{N} Z_i$.

Appendix B Proofs

Lemma B.1 *Given a simple graph $G = G(V, E)$ with $|V| = n$, and let $\overline{G} = G(V, \overline{E})$ be its complement. Let $\boldsymbol{x} \in \{0,1\}^n$ be a binary n-tuple such that its associated vertex set is a vertex cover in G; i.e., $V(\boldsymbol{x}) \in \mathscr{G}^{\text{vc}}$, and denote its complement by $\overline{\boldsymbol{x}}$. Furthermore, $f(\cdot)$ and $g(\cdot)$ are PMF's on binary n-tuples given by* (16) *and* (17)*, respectively. Then $f(\boldsymbol{x}) = g(\overline{\boldsymbol{x}})$.*

Proof. Note that $V(\overline{\boldsymbol{x}})$ is a clique in the complement graph $\overline{G}$.

First we show that $f(1) = g(0)$. This follows directly from their definitions:

$$f_1(1) = \frac{E_{k_+}}{E_{k_+} + E_{k_-}},$$

where $k_+ = n - 1$, and $k_- = d_1$ (degree of vertex v_1 in G), see Definition 2.1. And

$$g_1(0) = \frac{E_{m_-}}{E_{m_-} + E_{m_+}},$$

where $m_- = n - 1$ and $m_+ = d_1$, see Definition 3.2.

Consider the vertex v_i. The following cases are of interest.

1. Suppose that in graph G there exists $v_j, j < i$, such that $x_j = 0$ and $(v_j, v_i) \in E$. This means that v_i is adjacent to a certain vertex that is not part of the vertex cover. Thus, v_i must be chosen with probability 1; i.e., $f_i(1|x_1, \ldots, x_{i-1}) = 1$. Then, in the complement graph $\overline{x}_j = 1$ and $(v_j, v_i) \notin \overline{E}$. This means v_i is not adjacent to a certain vertex that is part of the clique. Thus, v_i is not chosen with probability 1; i.e., $g_i(0|\overline{x}_1, \ldots, \overline{x}_{i-1}) = 1$.

2. Suppose that in graph G all vertices $v_j, j < i$ that are are part of the vertex

cover, are adjacent to v_i. Mathematically:

$$x_j = 1 \ (j \leq i-1) \quad \Rightarrow \quad (v_j, v_i) \in E.$$

Thus v_i may or may be not part of the vertex cover in G. Let

$$A = \{v_k | k \geq i+1; \exists\, j \leq i-1 \ x_j = 0 \text{ and } (v_j, v_k) \in E\},$$
$$B = \{v_k | k \geq i+1; v_k \notin A; \text{and } (v_i, v_k) \in E\}.$$

Then v_i is chosen in the vertex cover with probability

$$f_i(1|x_1, \ldots, x_{i-1}) = \frac{E_{k_+}}{E_{k_+} + E_{k_-}},$$

with, according to Definition 2.1:

$$k_+ = n - i - |A|; \quad k_- = n - i - |A| - |B|.$$

In the complement graph $\overline{G}$ it holds that

$$\overline{x}_j = 0 \ (j \leq i-1) \quad \Rightarrow \quad (v_j, v_i) \notin \overline{E}.$$

Thus v_i may or may be not part of the clique in $\overline{G}$. Let

$$\overline{A} = \{v_k | k \geq i+1; \exists\, j \leq i-1 \ \overline{x}_j = 1 \text{ and } (v_j, v_k) \notin \overline{E}\},$$
$$\overline{B} = \{v_k | k \geq i+1; v_k \notin \overline{A} \text{ and } (v_i, v_k) \in \overline{E}\}.$$

Then v_i is not chosen in the clique with probability

$$g_i(0|\overline{x}_1, \ldots, \overline{x}_{i-1}) = \frac{E_{m_-}}{E_{m_-} + E_{m_+}},$$

with, according to Definition 3.2:

$$m_- = n - i - |\overline{A}|; \quad m_+ = |\overline{B}|.$$

We see immediately that the set A and $\overline{A}$ are the same, and that $\overline{B} = \{v_{i+1}, \ldots, v_n\} \setminus (\overline{A} \cup B)$. In other words, $k_+ = m_-$ and $k_- = m_+$.

□

Lemma B.2 *Given a simple graph $G = G(V, E)$ with $|V| = n$, and let $\overline{G} = G(V, \overline{E})$ be its complement. Let $\boldsymbol{x} \in \{0, 1\}^n$ be a binary n-tuple such that its associated vertex set is a clique in G; i.e., $V(\boldsymbol{x}) \in \mathscr{G}^{\mathrm{cl}}$, and denote its complement by $\overline{\boldsymbol{x}}$. Furthermore, $f(\cdot)$ and $g(\cdot)$ are PMF's on binary n-tuples given by* (16) *and* (17)*, respectively. Then $g(\boldsymbol{x}) = f(\overline{\boldsymbol{x}})$.*

The proof goes similarly as the proof of the previous Lemma, and therefore it is omitted.