On the Analysis of Independent Sets via Multilevel Splitting

Radislav Vaisman Dirk P. Kroese

December 22, 2017

Abstract

Counting the number of independent sets is an important problem in graph theory, combinatorics, optimization, and social sciences. However, a polynomial-time exact calculation, or even a reasonably close approximation, is widely believed to be impossible, since their existence implies an efficient solution to various problems in the non-deterministic polynomial-time complexity class. To cope with the approximation challenge, we express the independent set counting problem as a rare-event estimation problem. We develop a multilevel splitting algorithm which is generally capable of delivering accurate results, while using a manageable computational effort, even when applied to large graphs. We apply the algorithm to both counting and optimization (finding a maximum independent set) problems, and show that it compares favourably with the existing state of the art.

Keywords— Independent sets, Networks, Counting problem, Rare-event simulation, Multilevel Splitting, Markov chain Monte Carlo, Sequential importance sampling

1 Introduction

Given a finite undirected graph $G = (V, E)$ with vertex set V and edge set E, a k-cardinality *independent set* (k-IS) is a vertex subset $\mathcal{I} \subseteq V$ such that $|\mathcal{I}| = k$ and no two vertices from $\mathcal{I}$ are adjacent in G. A *maximum independent set* (max–IS) is defined as the largest possible independent set for a given graph G. The max-IS cardinality is called the graph's *independence number* and denoted by $\alpha(G)$ [24]. In this paper, we consider the independent set counting problem (#IS). Namely, given G and a set $\mathcal{K} \subseteq \{0, \ldots, |V|\}$, we ask for the total number of k-IS's in G for all $k \in \mathcal{K}$. In particular, we distinguish between three special cases of the #IS problem. Namely, the #k-IS problem, which is defined for a specific $0 \leq k \leq |V|$; the #all-IS problem, for counting *all* independent sets of G, note that in this case — $\mathcal{K} = \{0, \ldots, |V|\}$; and the max-IS problem defined above. It is worth noting that the problem of finding a graph's independence number is a special case of the #IS counting problem, since one can ask if there

exists an independent set of cardinality k for each $0 \leq k \leq |V|$, and thus find the corresponding $\alpha(G)$.

While #IS is a fundamental problem in graph theory and theoretical computer science [32], it also has many real-life applications. In particular, finding the distribution of different-cardinality independent sets in G is equivalent to counting the number of cliques in G's complement graph. These are of great interest to the analysis of social networks [1, 21, 34]. Under this setting, a group of individuals that share the same interests can be modelled by a corresponding clique in a graph. If, for example, one would like to target such groups, say via an advertisement or other social activity, a full-enumeration procedure of corresponding cliques is of great importance. However, such a routine can be extremely time consuming if the number of these cliques is large. Thus, knowledge of the distribution of the number of different-sized cliques can be very beneficial for determining the computational effort of a full-enumeration procedure. A similar idea of using counting estimators to probe the efficacy of backtracking algorithms is discussed in a seminal paper of [35].

In addition, the ubiquitous SAT (satisfiability) problem is a special case of #IS [16, 37, 50]. In particular, SAT can be reduced to the problem of finding the independence number $\alpha(G)$ of an induced graph G. The SAT problem is extensively used for handling important search and optimization problems in both computer science and operational research. In particular, some of the common problem domains are automated planning and scheduling [42, Chapter 7], automatic test pattern generation [39], and pipelined microprocessors [10]. Thus, the problem of finding a graph's independence number, can be of interest to operational research and first-order logic communities.

The #IS counting problem belongs to the #P complexity class introduced in [53], so an exact polynomial-time solution will imply P = NP [51]. Despite some recent advances [55], the #IS problem remains hard for graphs with maximum degree $\Delta > 5$. In fact, it was shown in [49] that there is no polynomial time approximation scheme for approximately counting independent sets on graphs of maximum degree $\Delta = 6$, improving the previous bound of $\Delta = 24$. Unfortunately, even a restricted case — the max-IS, is NP-hard [40]. Today, the most efficient exact algorithm for solving the max-IS problem runs in $\mathcal{O}\left(1.2127^{|V|}\right)$ time [9]. Moreover, for a general graph G, $\alpha(G)$ cannot be approximated to a constant factor in polynomial time, unless P = NP [5].

There exist two basic approaches for handling #P counting problems: (1) Markov Chain Monte Carlo (MCMC) [8, 30, 44, 45] and (2) Sequential Importance Sampling (SIS) [33, 15, 6, 7, 14, 47]. While MCMC methods dominate the field of counting approximations, the SIS-based approach can be a very powerful alternative. For example, [52] introduced a probabilistic relaxation technique, which is used to construct a set of good importance probabilities using Dynamic programming (DP). For the problem of counting all independent sets (#all-IS) in a graph, this specialized SIS algorithm was shown to outperform the existing state of the art techniques of [25, 26, 48] for counting the number of solutions in general conjunctive normal form (CNF) formulas.

Here we develop a much more flexible approach for the solution of the #IS problem, in the sense that it allows the handling of a variety of counting and optimization tasks; namely, #k-IS, #all-IS, and max-IS. The technique is based on Multilevel Splitting (MS) [31, 23, 8], and it is suitable for rare-event probability estimation. The latter is shown to be equivalent to the #IS counting problem in Section 2.

The main splitting idea can be briefly summarized as follows. Given a state space $\mathcal{X}$ and a subset of interest $\mathcal{X}^* \subseteq \mathcal{X}$ (such that $|\mathcal{X}^*| \ll |\mathcal{X}|$), find a performance function from $\mathcal{X}$ to $\mathbb{R}$ that partitions the space into decreasing cardinality level-sets. Then, the problem of sampling a rare $\mathbf{X} \in \mathcal{X}^*$, becomes one of generating a succession of events that are not rare. Namely, we initiate a random walk on these level-sets, which starts at $\mathcal{X}$ and ends at the smallest one ($\mathcal{X}^*$), with high probability. This stochastic process is generated using an MCMC sampler, which reproduces (splits) particles from each level set. While a brief review of the MS algorithm is given in Section 3, it is worth noting that MS was generalized in [8] to approximate a wide range of rare-event, optimization and counting problems. For a comprehensive review, we refer the reader to [38, 43] and [47, Chapter 4].

In this paper we develop an adaptation of the general MS algorithm for the #IS problem. The proposed method is computationally efficient, enjoys a good practical performance and, similar to [27], has a probabilistic lower bound on its output. The latter allows us to reliably handle large graph instances.

The rest of the paper is organized as follows. In Section 2 we formulate the #IS counting problem through a random sampling procedure and explain the corresponding rare-event probability estimation. In Section 3 we give a brief introduction to the MS algorithm, show that it can be applied to all types of #IS counting problems (#all-IS, #k-IS, and max-IS), and describe a set of probabilistic lower bounds. We report our numerical findings in a detailed experimental analysis in Section 4. To do so, a freely available research software package called IsSplit was developed. Our benchmark study indicates the following.

1. For the problem of counting all independent sets, the proposed MS algorithm outperforms the SIS procedure of [52] for non-random graphs.

2. Combining the MS method with a simple binary search can be used to solve the max-IS problem and, consequently, to find a graph's independence number.

3. Probabilistic lower bounds (given in Section 3) allow a faster approximation of the distribution of k-IS's, and thus save considerable computation effort. The latter allows the analysis of non-trivial graph instances with hundreds of vertices. To the best of our knowledge, there exists no method today that is able to perform this task on large graphs.

Finally, in Section 5 we summarize our findings and discuss possible directions for future research.

2 Problem formulation

To start with, we examine an important object which gives a full specification of the number of independent sets in a graph — the Independence Polynomial (IP) [29]. The IP is a useful indicator of how difficult it is to estimate the number of independent sets. We also find it helpful for benchmarking the MS algorithm.

2.1 The independence polynomial

The IP takes the form

$$\mathcal{IP}(G, x) = \sum_{k=0}^{\alpha(G)} s_k \, x^k,$$

where s_k is the number of independent sets of cardinality k in G, and $\alpha(G)$ is the independence number.

Although the #IS problem is generally hard, we know the exact IP analytically for some specific graph topologies. One such graph is called the n-Andrásfai graph ($\mathcal{A}_n$) [2], which is a circulant graph on $3n-1$ nodes. The $\mathcal{A}_n$'s edges are defined through the vertex ordering $(v_1, \ldots, v_{3n-1})$, such that for any two vertices v_i, v_j with $1 \leq i < j \leq 3n-1$, an edge between v_i and v_j exists if $(j-i) \mod 3 = 1$. An example $\mathcal{A}_4$ graph is shown in Figure 1.

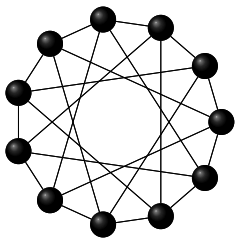


Figure 1: The $\mathcal{A}_4$ graph.

While various attractive properties of the $\mathcal{A}_n$ graph are available in [24, Chapter 6.10-6.12], our interest is focused on its IP, which is equal to [41]

$$\mathcal{IP}(\mathcal{A}_n, x) = 1 + (3n-1)x(x+1)^{n-1}. \tag{1}$$

Next, we reduce the #IS counting problem to the problem of estimating the probability of selecting an independent set uniformly at random.

2.2 The probabilistic set-up

Let $V = \{v_1, \ldots, v_n\}$ be a graph's vertex set and let $[V]^k := \{A \subseteq V \ : \ |A| = k\}$ be the set of all vertex subsets having cardinality k. Consider the uniform distribution on the $[V]^k$ set, and let $\mathbf{X} = \{X_1, \ldots, X_k\}$ be a uniform random assignment from $[V]^k$. Finally, let ℓ be the probability that $\mathbf{X}$ is an independent set. Then, the number of k-IS's, (s_k), is equal to $\binom{n}{k}\ell$. Thus, estimation of

s_k can be achieved through estimation of ℓ. The Crude Monte Carlo (CMC) procedure for the estimation of ℓ is summarized in Algorithm 1.

Algorithm 1 CMC Algorithm for estimating ℓ

Input: A graph $G = (V, E)$, $1 \leq k \leq |V|$, and a sample size $N \in \mathbb{N}$.
Output: Unbiased estimator of ℓ.

```
for t = 1 to N do
    Choose X_t uniformly at random from [V]^k.
    if X_t is an independent set in G then
        Y_t = 1,
    else
        Y_t = 0.
    end if
end for
return: ℓ̂_CMC = N^{-1} Σ_{t=1}^{N} Y_t.
```

Unfortunately, when working with a general graph, ℓ can be very small (say less than 10^{-10}). In this case, this probability is so tiny that for a reasonable sample size N (such as $N \approx 10^6$), the random variables $Y_1, \ldots, Y_N$ are all 0 with high probability, and the resulting estimator in Algorithm 1 is thus $\hat{\ell}_{\text{CMC}} = 0$.

This CMC issue is well known and is called the rare-event probability estimation problem [3, 46]. Under the rare-event setting, a main measure of algorithmic efficiency for an unbiased estimator $\hat{\ell}$ is the Relative Error (RE), which is defined [43] by $\text{RE} = \frac{\sqrt{\text{Var}(\hat{\ell})}}{\mathbb{E}[\hat{\ell}]}$, also called the coefficient of variation (CV).

Consider an $\mathcal{A}_{15}$ graph as an example of a graph that falls into the rare-event trap when applying the CMC procedure in Algorithm 1. The graph's IP in (1) will reveal that there are 44 independent sets of cardinality 15. However, since the $\mathcal{A}_{15}$ graph has 44 vertices, $\ell_{15} = 44/\binom{44}{15} \approx 1.91 \times 10^{-10}$. From this, we can calculate the sample size (N) that is required to reach a modest 10% RE. In particular, as $\text{RE} \approx 1/\sqrt{N\ell}$ for small ℓ, the sample size should satisfy $N \geq 5.23 \times 10^{11}$.

While such an N is clearly unmanageable from the perspective of the required computation effort, we overcome this issue by adapting the MS algorithm of [8] for the #IS counting problems. The general splitting framework is discussed next.

3 Multilevel splitting

We consider the following setting. Let $\mathcal{X}$ and $\mathcal{X}^* \subseteq \mathcal{X}$ be sets such that $|\mathcal{X}^*| \ll |\mathcal{X}|$, and suppose that the probability $\ell = |\mathcal{X}^*|/|\mathcal{X}|$ is to be estimated. Recall that the main idea of MS is to design a sequential sampling plan that decomposes the "difficult" problem of sampling from $\mathcal{X}^*$ into a number of "easy" ones associated with a sequence of subsets in the $\mathcal{X}$ sampling space. A general

MS framework is summarized below.

1. *Level-sets.* Find a sequence of sets $\mathcal{X} = \mathcal{X}_0 \supseteq \cdots \supseteq \mathcal{X}_T = \mathcal{X}^*$, and define a performance function $S : \mathcal{X} \to \mathbb{R}$ such that the subsets $\mathcal{X}_t$ can be written as level sets of S for levels

$$\infty = \gamma_0 \geq \gamma_1 \geq \cdots \geq \cdots \geq \gamma_T = \gamma, \tag{2}$$

that is, $\mathcal{X}_t = \{\mathbf{x} \in \mathcal{X} \,:\, S(\mathbf{x}) \leq \gamma_t\}$ for $t = 0, \ldots, T$. Then, ℓ is given by

$$\ell = \frac{|\mathcal{X}^*|}{|\mathcal{X}|} = \prod_{t=1}^{T} \frac{|\mathcal{X}_t|}{|\mathcal{X}_{t-1}|} = \prod_{t=1}^{T} \mathbb{P}\left(S\left(\mathbf{X}\right) \leq \gamma_t \mid S\left(\mathbf{X}\right) \leq \gamma_{t-1}\right) = \mathbb{P}\left(S(\mathbf{X}) \leq \gamma_T\right).$$

2. *Conditional probability estimation.* For each $t = 1, \ldots, T$, develop an efficient estimator $\hat{c}_t$ for the conditional probabilities:

$$c_t = \mathbb{P}\left(S(\mathbf{X}) \leq \gamma_t \mid S(\mathbf{X}) \leq \gamma_{t-1}\right). \tag{3}$$

To avoid the rare-event problem at the intermediate levels $t = 1, \ldots, T$, we assume that the sets $\{\mathcal{X}_t\}$ are specifically designed such that the $\{c_t\}$ are not too small, that is, c_t is not a rare-event probability.

To put the #IS problem into the MS framework, we must address the definition of the level sets and the procedure of estimating the conditional probabilities in (3).

1. *The #IS level-sets.* Given a graph G, let $\mathcal{X}$ be the set of all vertex subsets, and let $\mathcal{X}^*$ be the set of all *independent* vertex subsets. In this paper we define $S(\mathbf{x})$ as *the number of adjacent vertices* in $\mathbf{x} \in \mathcal{X}$. For this performance function, it holds (see the level-sets definition above) that $\mathcal{X}_0 = \{\mathbf{x} \in \mathcal{X} \,:\, S(\mathbf{x}) \leq \infty\}$, and $\mathcal{X}^* = \{\mathbf{x} \in \mathcal{X} \,:\, S(\mathbf{x}) = 0\}$, respectively.

2. *The #IS conditional probability estimation.* For each $t = 2, \ldots, T$, we estimate the conditional probability c_t in (3) via N uniform random samples from $\mathcal{X}_{t-1}$. In particular, the estimate $\hat{c}_t$ is equal to the number of these samples that fall into $\mathcal{X}_t$ divided by N. This uniform sampling on the $\mathcal{X}_{t-1}$ set is achieved in Algorithms 3 and 4 via the MCMC Gibbs sampler [11, 8, 47].

The general MS procedure (which provides an unbiased estimator of ℓ [8]) is given in Algorithm 2.

Remark 1 (Performance function selection)**.** *It is important to note that a particular choice of performance function can have a crucial impact on the performance of the MS algorithm (see Example 4.5 in Section 4). We refer to [18, 19] for additional discussion.*

Note that the cardinality parameter k in Algorithm 2 is marked as optional. The latter is due to the fact that the MS algorithm is capable of operating in two counting modes. Namely, it can perform a counting estimation of both #k-IS and #all-IS. Naturally, when we solve the #all-IS problem, the parameter k is irrelevant. The counting mode is controlled via the corresponding Markov transition step of Algorithm 2 (line 8), and is discussed next in detail.

Algorithm 2 MS Algorithm for estimating ℓ

Input: A graph $G = (V, E)$, $1 \leq k \leq |V|$ (optional), a sequence of levels $\gamma_1, \ldots, \gamma_T$, a performance function $S : \mathcal{X} \to \mathbb{R}$, and a sample size $N \in \mathbb{N}$.

Output: Unbiased estimator of ℓ.

Initialization: Generate N independent samples $\mathcal{V}_0 = \{\mathbf{X}_1, \ldots, \mathbf{X}_N\}$ uniformly from $\mathcal{X}$. Let $\mathcal{W}_1 \subseteq \mathcal{V}_0$ be the subset of elements $\mathbf{X}$ in $\mathcal{V}_0$ for which $S(\mathbf{X}) \leq \gamma_1$ holds (that is, $\mathcal{W}_1$ is an elite population of particles), let N_1 be the size of $\mathcal{W}_1$, and set $\hat{c}_1 = N_1/N$.
for $t = 1$ **to** $T - 1$ **do**
  Draw $K_i \sim \mathsf{Bernoulli}(0.5)$, for $i = 1, \ldots, N_t$, such that $\sum_{i=1}^{N_t} K_i = N \bmod N_t$.
  for $i = 1$ **to** N_t **do**
    Set a splitting factor $S_{ti} = \lfloor N/N_t \rfloor + K_i$, where $\lfloor x \rfloor$ is the floor function. Note that K_i is just a random number (0 or 1) that ensures that the $\mathcal{V}_t$ set contains exactly N samples, i.e., that $\sum_{i=1}^{N_t} S_{ti} = N$ holds.
    Set $\mathbf{Y}_{i,0} = \mathbf{X}_i$.
    for $j = 1$ **to** S_{ti} **do**
      Draw $\mathbf{Y}_{i,j} \sim \kappa_t(\mathbf{y} \mid \mathbf{Y}_{i,j-1})$, where κ_t is a Markov transition density whose stationary distribution is the uniform distribution on $\mathcal{X}_t$ (we describe this in Section 3.1).
    end for
  end for
  Set the population

$$\mathcal{V}_t = \{\mathbf{Y}_{1,1}, \ldots, \mathbf{Y}_{1,S_{t1}}, \cdots, \mathbf{Y}_{N_t,1}, \ldots, \mathbf{Y}_{N_t,S_{tN_t}}\}.$$

  Note that $\mathcal{V}_t$ contains N elements.
  Let $\mathcal{W}_{t+1} \subseteq \mathcal{V}_t$ be the subset of elements of $\mathcal{V}_t$ for which $S(\mathbf{Y}) \leq \gamma_{t+1}$, and let N_{t+1} be the size of new population $\mathcal{W}_{t+1}$.
  Estimation: Set $\hat{c}_t = N_{t+1}/N$.
end for
return: $\hat{\ell} = \prod_{t=1}^{T} \hat{c}_t$.

3.1 The Gibbs sampler

Algorithms 3 and 4 introduce the MCMC steps that are required for handling the #k-IS and #all-IS problems, respectively.

Remark 2 (Sampling considerations)**.** *Ideally, at each step $1 \leq t \leq T$, we would like to sample from the uniform probability density function (pdf) $p_{\gamma_t}(\mathbf{x})$ defined on $\{\mathbf{x} \in [V]^k \ : \ S(\mathbf{x}) \leq \gamma_t\}$ and the uniform pdf $g_{\gamma_t}(\mathbf{x})$ defined on $\{\mathbf{x} \subseteq V \ : \ S(\mathbf{x}) \leq \gamma_t\}$. Since such a direct generation is hard, we resolve this by using the corresponding Gibbs samplers. In particular, Algorithm 3 samples consecutively from the conditional pdfs $p_{\gamma_t}(x_i \mid x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_k)$, $1 \leq i \leq k$. The latter is not very difficult to sample from, since it only requires one to generate $X_i \in V \setminus \{x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_k\}$ uniformly at random, subject to the constraint that the resulting $\mathbf{X} = (x_1, \ldots, x_{i-1}, X_i, x_{i+1}, \ldots, x_k)$ lies in $\{\mathbf{x} \in [V]^k \ : \ S(\mathbf{x}) \leq \gamma_t\}$. In other words, we require that $S(\mathbf{X}) \leq \gamma_t$ holds. Moreover, in order to implement the Gibbs sampling in Algorithm 4, we generate*

from the conditional pdf $g_{\gamma_t}(\mathbb{1}_{\{v_i \in \mathbf{x}\}} \mid \mathbf{x})$, $1 \leq i \leq |V|$. *Namely, given* $\mathbf{x}$, *we form two sets:* $\tilde{\mathbf{X}}' = \mathbf{x} \cup \{v_i\}$, *and* $\tilde{\mathbf{X}}'' = \mathbf{x} \setminus \{v_i\}$. *If both* $\tilde{\mathbf{X}}'$ *and* $\tilde{\mathbf{X}}''$ *belong to* $\{\mathbf{x} \subseteq V \ : \ S(\mathbf{x}) \leq \gamma_t\}$, *we accept each with probability* $1/2$. *Otherwise, we accept* $\tilde{\mathbf{X}}''$, *since it always holds that* $S(\tilde{\mathbf{X}}'') \leq \gamma_t$.

Algorithm 3 Gibbs sampler for sampling uniformly from the $\mathcal{X}_t$ set in the #k-IS mode.

Input: A graph $G = (V, E)$, a threshold value γ_t, and an element $\mathbf{X} \in [V]^k$ such that $S(\mathbf{X}) \leq \gamma_t$.
Output: $\tilde{\mathbf{X}}$ distributed approximately uniformly on $\mathcal{X}_t$.

Set $\tilde{\mathbf{X}} = \mathbf{X}$. (Optionally, set $\tilde{\mathbf{X}}$ to be a random permutation of $\mathbf{X}$.)
for $i = 1$ **to** k **do**
  Set $V' = V \setminus \tilde{\mathbf{X}}$, $A = \{X_i\}$, $\tilde{\mathbf{X}} = \tilde{\mathbf{X}} \setminus \{X_i\}$.
  for all $v \in V'$ **do**
    if $S\left(\tilde{\mathbf{X}} \cup \{v\}\right) \leq \gamma_t$ **then**
      $A = A \cup \{v\}$.
    end if
  end for
  Choose $\tilde{X}_i$ uniformly at random from A (with probability $1/|A|$), and set $\tilde{\mathbf{X}} \cup \{\tilde{X}_i\}$.
end for
return $\tilde{\mathbf{X}}$.

Algorithm 4 Gibbs sampler for sampling uniformly from the $\mathcal{X}_t$ set in the #all-IS mode.

Input: A graph $G = (V, E)$, a threshold value γ_t, and an element $\mathbf{X} \in \mathcal{X}$, such that $S(\mathbf{X}) \leq \gamma_t$.
Output: $\tilde{\mathbf{X}}$ distributed approximately uniformly on $\mathcal{X}_t$.

Set $\tilde{\mathbf{X}} = \mathbf{X}$.
for $i = 1$ **to** $|V|$ **do**
  Set $\tilde{\mathbf{X}}' = \tilde{\mathbf{X}} \cup \{v_i\}$ and $\tilde{\mathbf{X}}'' = \tilde{\mathbf{X}} \setminus \{v_i\}$.
  if $S(\tilde{\mathbf{X}}') > \gamma_t$ **and** $S(\tilde{\mathbf{X}}'') \leq \gamma_t$ **then**
    Set $\tilde{\mathbf{X}} = \tilde{\mathbf{X}}''$.
  end if
  if $S(\tilde{\mathbf{X}}') \leq \gamma_t$ **and** $S(\tilde{\mathbf{X}}'') \leq \gamma_t$ **then**
    Set $\tilde{\mathbf{X}} = \tilde{\mathbf{X}}'$ or $\tilde{\mathbf{X}} = \tilde{\mathbf{X}}''$ with equal probability.
  end if
end for
return $\tilde{\mathbf{X}}$.

The computational complexities of Algorithms 3 and 4 are discussed in Proposition 1.

Proposition 1 (Computational implementation)**.** *For a given graph* $G = (V, E)$, *Algorithm 3 and Algorithm 4 can be completed in* $\mathcal{O}(k|V||E|)$ *and* $\mathcal{O}(|E|)$ *time, respectively.*

Proof. Given a graph $G = (V, E)$ and a subset of vertices $\mathbf{X} \subseteq V$, define $\mathrm{Ng}(v, \mathbf{X})$ (for any $v \in V$) to be the set of all vertices in $\mathbf{X}$ that are adjacent to v; that is,

$$\mathrm{Ng}(v, \mathbf{X}) = \{u \ : \ (v, u) \in E, \ u \in \mathbf{X}\}.$$

Then, $S(\mathbf{X} \cup \{v \notin \mathbf{X}\})$ and $S(\mathbf{X} \setminus \{v \in \mathbf{X}\})$ are equal to $S(\mathbf{X}) + 2\,|\mathrm{Ng}(v, \mathbf{X})|$ and $S(\mathbf{X}) - 2\,|\mathrm{Ng}(v, \mathbf{X})|$, respectively. The latter holds, since when we add (remove) a vertex v to (from) $\mathbf{X}$, the corresponding performance is increased (decreased) by 2 multiplied by the number of $v's$ adjacent vertices that are in $\mathbf{X}$. Moreover, provided that $S(\mathbf{X})$ is known and that G is given by an adjacency matrix, the calculation of $S(\mathbf{X} \cup \{v\})$ and $S(\mathbf{X} \setminus \{v\})$, can be performed in $\mathcal{O}(\deg(v))$ time, where $\deg(v)$ stands for $v's$ degree. Combining this with the fact that, for any $G = (V, E)$, $\sum_{v \in V} \deg(v) = 2|E|$ holds, we complete the proof by noting that the time complexities of Algorithms 3 and 4 are given by $k\,\mathcal{O}\left(\sum_{v \in V} \deg(v)\right) = \mathcal{O}(k\,|E|)$ and $\sum_{v \in V} \deg(v) = \mathcal{O}(|E|)$, respectively. □

Proposition 1 opens a way for the time complexity analysis of the MS Algorithm 2. In particular, the initialization step can be performed in $\mathcal{O}(N|E|)$ time and the MCMC step is performed for $\mathcal{O}(NT)$ times during the algorithm execution. Hence, the overall complexity is equal to $\mathcal{O}(kTN|E|)$ and $\mathcal{O}(TN|E|)$ for $\#k$-IS and #all-IS counting modes, respectively.

The above computational complexity consideration is of practical importance. The next section discusses the asymptotic efficiency of Algorithm 2.

3.2 The MS algorithm efficiency

There exists a well-known asymptotic result [20, 12, Proposition 3], stating that for quite general splitting estimators $\hat{\ell}$

$$\sqrt{N}\,\frac{\hat{\ell} - \ell}{\ell} \xrightarrow[N \to \infty]{\mathcal{D}} \mathcal{N}(0, \sigma^2),$$

where σ^2 depends on the complicated way on the dependence structure of the particles in the $\{\mathcal{V}_t\}_{0 \le t \le T-1}$ sets. Under an idealized setting of particle independence (also considered in Theorem 1 below), σ^2 simplifies to $\sigma^2 = \sum_{t=1}^{T} (1 - c_t)/c_t$ [12].

In this paper, we consider the MS algorithm efficiency via the squared coefficient of variation of the output $\hat{\ell}$; that is $\mathrm{CV}^2 = \mathrm{Var}\left(\hat{\ell}\right) / \left[\mathbb{E}\,\hat{\ell}\right]^2$. Following [33], a randomized algorithm is considered efficient, if its output CV^2 is bounded by a polynomial in the algorithm input. Such an algorithm is called a fully polynomial randomized approximation scheme and this result is basically the best one can hope to achieve when dealing with #P problems such as #IS, for which an efficient analytical solution is not known to exist.

Although one cannot expect to show such a result for a general #IS problem, Algorithm 2 can be analyzed under some simplified independence assumptions. In particular, we obtain an exact result about the MS's estimator in the idealized

situation of [12] where each step begins with independent particles. This would typically correspond to the situation where at each step the Gibbs sampler is applied an infinite number of times. The theoretical result below provides some insight into the working of the MS algorithm.

Theorem 1 (Efficiency of MS Algorithm 2)**.** *Suppose that all particles in $\mathcal{V}_t$, $0 \le t \le T-1$, are independent of each other and that $c_t = \mathcal{O}(1/\mathcal{P}_T)$, where $\mathcal{P}_T$ is a polynomial in T. Then, the MS algorithm is computationally efficient [33]; in particular,* $\mathrm{CV}^2 = e - 1$.

Proof. The analysis is by obtaining the lower and the upper bounds for the first and the second moments of $\hat{\ell}$, respectively.

1. **First moment**. From the unbiasedness of the MS algorithm,

$$\mathbb{E}\left[\hat{\ell}\right] = \ell = \prod_{t=1}^{T} c_t,$$

and hence

$$\left(\mathbb{E}\left[\hat{\ell}\right]\right)^2 = \left(\prod_{t=1}^{T} c_t\right)^2 = \prod_{t=1}^{T} c_t^2. \tag{4}$$

2. **Second moment**. By the theorem assumption, the samples in $\mathcal{V}_{t-1}$ are independent for all $t = 1, \ldots, T$, and hence the (random) cardinalities of $\mathcal{W}_t$ sets are binomially distributed according to $N_t \sim \mathsf{Bin}(N, c_t)$. Also, $\{\hat{c}_t\}$ are independent. Thus, for all $t = 1, \ldots, T$,

$$\mathbb{E}\left[\hat{c}_t^2\right] = \mathbb{E}\left[\left(\frac{N_t}{N}\right)^2\right] = \left(\frac{c_t(1-c_t)}{N} + c_t^2\right), \tag{5}$$

and letting $c_{\min} = \min_{1 \le t \le T}\{c_t\}$, we arrive at

$$\begin{aligned}
\mathbb{E}\left[\hat{\ell}^2\right] = \mathbb{E}\left[\left(\prod_{t=1}^{T} \hat{c}_t\right)^2\right] &= \prod_{t=1}^{T} \mathbb{E}\left[\hat{c}_t^2\right] \underbrace{=}_{(5)} \prod_{t=1}^{T}\left(\frac{c_t(1-c_t)}{N} + c_t^2\right) \qquad (6)\\
&= \prod_{t=1}^{T} c_t^2 \left(1 + \frac{(1-c_t)}{N c_t}\right) \le \prod_{t=1}^{T} c_t^2 \prod_{t=1}^{T}\left(1 + \frac{1}{N c_t}\right)\\
&\le \left(1 + \frac{1}{N c_{\min}}\right)^T \prod_{t=1}^{T} c_t^2 \underbrace{\le}_{N c_{\min} \ge T} e \prod_{t=1}^{T} c_t^2,
\end{aligned}$$

where the last inequality follows from the well-known identity

$$(1 + 1/n)^n \le e, \quad n > 0.$$

Combining (4) and (6) yields

$$\mathrm{CV}^2 = \frac{\mathrm{Var}\left(\hat{\ell}\right)}{\left[\mathbb{E}\,\hat{\ell}\right]^2} = \frac{\mathbb{E}\left[\hat{\ell}^2\right]}{\left[\mathbb{E}\,\hat{\ell}\right]^2} - 1 \leq \frac{e\,\prod_{t=1}^{T} c_t^2}{\prod_{t=1}^{T} c_t^2} - 1 = e - 1, \tag{7}$$

and we complete the proof by noting that the required sample size N is polynomial in T, since (7) holds for $N = \lceil T/c_{\min} \rceil = \mathcal{O}(\mathcal{P}_T)$. □

We proceed with a simple idea that allows Algorithm 2 to discover a graph's independence number. The proposed adaptation is a straightforward binary search on the independent set cardinality $1 \leq k \leq |V|$. The procedure is summarized in Algorithm 5.

3.3 Finding a graph's independence number

Algorithm 5 Binary search algorithm for estimating the graph's independence number

Input: A graph $G = (V, E)$, and sample size $N \in \mathbb{N}$.
Output: An estimation of $\alpha(G)$.

1: Set low = 1 and high = $|V|$.
2: **while** low < high **do**
3: Set mid = $\lceil$low + high$\rceil/2$.
4: Let $\hat{\ell}_{\text{mid}}$ be the estimated number of independent sets of cardinality k in G which is obtained using the MS Algorithm 2.
5: **if** $\hat{\ell}_{\text{mid}} > 0$ **then**
6: high = mid.
7: **else**
8: low = mid.
9: **end if**
10: **end while**
11: Set $\hat{\alpha}(G)$ = mid.
12: **return** $\hat{\alpha}(G)$ as an approximation of $\alpha(G)$.

Noting that Algorithm 5 will execute the main MS method (Algorithm 2), for $\mathcal{O}\left(\log_2 |V|\right)$ times, and combining this with the MS running time, yields the overall time complexity of $\mathcal{O}\left(\log_2 \left(|V|\right) TN|V||E|\right)$.

Remark 3 (The solution to the max-IS problem)**.** It is worth noting that Algorithm 5 provides a solution to the max-IS problem, too. All we need to do is to record an $\alpha(G)$-cardinality independent set from the last execution of the MS Algorithm 2 in line 4 of the binary search procedure.

From a practical point of view, an interesting research question is to find the distribution of k-cardinality independent sets in a graph $G = (V, E)$ for $k = 1, \ldots, |V|$. Namely, to discover a good approximation to each coefficient of G's independence polynomial. Taking into consideration the computational

complexity of such a procedure for large graphs, a reasonable approach is to use a probabilistic lower bound, which was first introduced by [27], and is provided next.

3.4 Probabilistic lower bounds

When running a Monte Carlo algorithm which provides an unbiased estimator, one can obtain a probabilistic lower bound [27]. In particular, the following holds.

Theorem 2 (Probabilistic lower bounds)**.** *Let $Z_1, \ldots, Z_R$ be independent realizations of random variables Z such that $\mathbb{E}(Z) = \mu$. Then, for a constant $0 \leq \omega < 1$, the following probabilistic lower bounds exist.*

1. *Minimum scheme bound (MSB):*

$$\mathbb{P}\left(\min_{1\leq r\leq R}\left[\frac{Z_r}{\eta}\right] \leq \mu\right) \geq \omega, \quad \textit{where} \quad \eta = \left(\frac{1}{1-\omega}\right)^{\frac{1}{R}}.$$

2. *Average scheme bound (ASB):*

$$\mathbb{P}\left(\left[\frac{\frac{1}{R}\sum_{r=1}^{R} Z_r}{\eta}\right] \leq \mu\right) \geq \omega, \quad \textit{where} \quad \eta = \frac{1}{1-\omega}.$$

3. *Maximum scheme bound (MASB):*

$$\mathbb{P}\left(\max_{1\leq r\leq R}\left[\frac{Z_r}{\eta}\right] \leq \mu\right) \geq \omega, \quad \textit{where} \quad \eta = \frac{1}{1-\omega^{\frac{1}{R}}}.$$

4. *Permutation scheme bound (PSB):*

$$\mathbb{P}\left(\max_{1\leq r\leq R}\left[\left(\frac{1}{\eta}\prod_{j=1}^{r} Z_j\right)^{1/r}\right] \leq \mu\right) \geq \omega, \quad \textit{where} \quad \eta = 1/(1-\omega).$$

5. *Order Statistics bound (OSB):*

$$\mathbb{P}\left(\max_{1\leq r\leq R}\left[\left(\frac{1}{\eta}\prod_{j=1}^{r}\frac{O_{(R-j+1)}}{\binom{R}{r}}\right)^{1/r}\right] \leq \mu\right) \geq \omega,$$

where $\eta = 1/(1-\omega)$, and $\left(O_{(1)}, \ldots, O_{(R)}\right)$ is an order statistic over the sample set $(Z_1, \ldots, Z_R)$, such that for $1 \leq r_1 < r_2 \leq R$, it holds that $O_{(r_1)} \leq O_{(r_2)}$.

Proof. See [27]. □

While these bounds are available after the first few independent runs of both Algorithm 2 and the SIS relaxation method of [52], one should consider the following limitation.

Remark 4 (Probabilistic lower bounds limitation)**.** The quality (tightness) of these bounds depends on the variance of the MS algorithm's output. As noted in [27], the MSB bound tends to decrease (unless the variance is very small) when the number of samples grows. The ASB and MASB schemes are based on the assumption that the sample average and maximum are likely to get larger as more samples are drawn. However, the problem with the MSB approach is that when the number of samples R increases, the η parameter increases as well, and thus the lower bound decreases. As an alternative to the MSB sceme, the PSB and the OSB bounds were introduced in [27].

Despite the above limitation, these bounds still introduce a very handy addition as we show in the numerical section. For example, our study indicates that for the MS algorithm these bounds are tight and are generally within an order of magnitude of the true result (or better) after only a few runs. A practical recipe for the usage of probabilistic lower bounds is summarized in Remark 5.

Remark 5 (Probabilistic lower bounds in practice)**.** Since the effort to calculate the bounds is relatively cheap compared to the overall central processing unit (CPU) time, we propose computing all the bounds from Theorem 2 and choosing the one that provides the best performance, namely, the one that introduces the highest lower bound. Then, we run the MS algorithm again and use the chosen (best-performing) bound. We call such a bound a Probabilistic Maximal Lower Bound (PMLB).

We complete this section by addressing two technical issues.

3.5 Implementation details

The MS algorithm parameters. The algorithm requires the specification of the sample size N and the intermediate performance function thresholds $\gamma_1, \ldots, \gamma_T$, which can be done as follows. The parameter N can be determined by trial and error, subject to the requirement that $\hat{c}_t \neq 0$ with high probability, for $t = 1, \ldots, T$. Also, to ensure the MS estimator's unbiasedness, it is crucial to fix the $\{\gamma_t\}$ levels in advance. To determine the levels, we perform a single *pilot run* of a slightly modified version of Algorithm 2 using a so-called rarity parameter ρ. The ρ parameter specifies the fraction of the number of elements in $\mathcal{V}_t$ that will proceed to the next level. Then, we fix γ_t to be equal to the worst performance value of these next-level samples. As a consequence, this pilot run helps us to establish a set of threshold values adapted to the specific graph; see [8] for further details. One should keep in mind that the success of the MS algorithm is dependent on the values of the conditional probabilities $\{c_t\}_{1 \leq t \leq T}$ (and thus the corresponding estimators $\{\hat{c}_t\}_{1 \leq t \leq T}$). In particular, the values of c_t should not be very small (see Example 4.5 in Section 4). In such a scenario,

the corresponding sample size N will be manageable (from a computational point of view), and the MS algorithm will be efficient as illustrated in the proof of Theorem 1. Finally, the MS algorithm success is not very sensitive to the particular choice of ρ. Nevertheless, ρ will affect the number of levels and thus the total execution time. In particular, the number of MS levels is equal to $\mathcal{O}\left(\lfloor \ln \ell / \ln \rho \rfloor\right)$ [13].

Controlling the MS algorithm error. A common practice when working with a Monte Carlo algorithm that outputs an unbiased estimator is to run it for R independent replications and report the average $\hat{\ell} = R^{-1} \sum_{r=1}^{R} \hat{\ell}_r$. In this paper, the quality of $\hat{\ell}$ is measured as follows.

- If ℓ is available (that is, we perform a benchmarking of the MS algorithm for an instance with a known IP) we use a Relative Experimental Error (RER) [17], which is defined by

$$\text{RER} = \left| \hat{\ell} - \ell \right| \cdot \ell^{-1}.$$

- If ℓ is not available (which is generally the case), the confidence interval is measured via the RE.

4 Numerical study

We investigate the accuracy of the MS algorithm and the SIS graph relaxation method [52], when applied to several representative graph instances, both random and non-random. The MS and the SIS algorithms were implemented in C++ packages called IsSplit and IsSIS, respectively; these software kits are freely available along with all examples from the author's web-page at `http://www.smp.uq.edu.au/people/RadislavVaisman/#software`. Our tests were executed on a Intel Core i7-3770 quad-core 3.4GHz processor with 8GB of RAM, using Windows 7 (64 bit version) and a `MSVC++ 11.0/1700` compiler version. Our implementation of all software packages is in a single-thread, though parallelization would be not very hard to add. The rest of the section is structured as follows.

1. The first graph under consideration is the Andrásfai $\mathcal{A}_{35}$ graph. Recall that its IP is known, so this example enables the precise benchmarking of the MS and the SIS algorithms accuracy.

2. To investigate the algorithms' performance on both random and non-random graphs, we consider Hypercube [28] and Waxman [54] graphs. These graphs provide a more realistic challenge, since we examine a few instances of different sizes, and thus the proposed methods' speed and accuracy can be judged.

3. Next, we consider four non-trivial benchmarks from `https://turing.cs.hbg.psu.edu/txn131/clique` and show that MS identifies their independence number correctly. One of these instances does not have an analytical

solution, as only a lower bound is known. Using the MS method, we conjecture that the known lower bound is equal to the true independence number.

4. We finish our numerical study with a cautionary example. In particular, we consider a very simple book graph [22], for which the MS algorithm will fail to solve the max-IS problem. While it is not very surprising that the MS method can fail for certain graph instances [40], we discuss the reason for this problem, and suggest a direction for a possible improvement.

MS parameters

We ran a number of preliminary benchmarks (not reported here) to determine reasonably robust parameter settings for N and ρ. The following parameters were used for all results described here.

- For the MS pilot run, we take $\rho = 20\%$.
- We set the sample size to be $N = 1000$, unless stated otherwise.
- In all experiments, we run R independent replications of the MS and the SIS methods, until the RE is smaller than or equal to a 3% threshold.
- In experiments where the lower bound is reported, we use the PMLB.

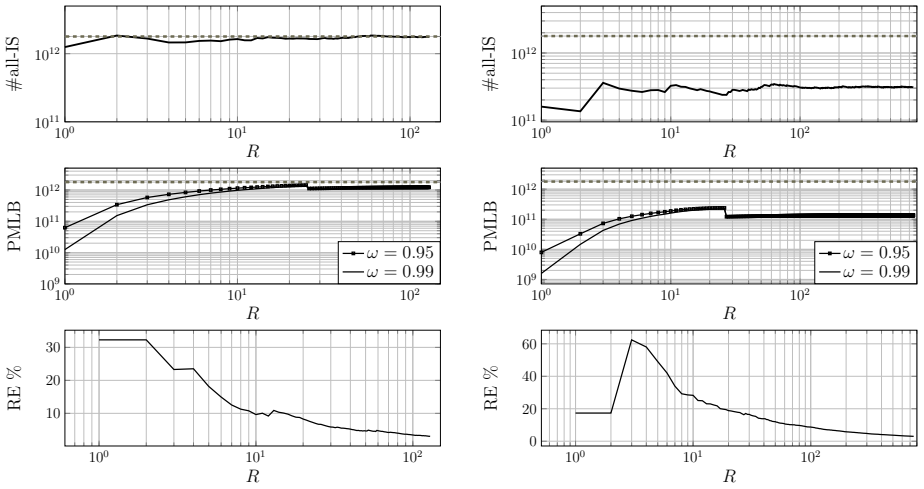
4.1 The 35-Andrásfai graph

In this example we are interested in an exact benchmarking of the proposed MS Algorithm 2 in the sense of RER, which is available via the $\mathcal{A}_{35}$'s IP in (1). Next, we apply the MS and the SIS algorithms for solving the #all-IS problem. Note that the SIS method cannot be adapted to #k-IS and max-IS, thus these problems are solved with the MS algorithm only.

Solving the #all-IS problem

Figure 2 summarizes the performance of MS and SIS for counting all independent sets of the $\mathcal{A}_{35}$ graph.

The SIS algorithm is faster. In particular, to reach the predefined 3% RE threshold we need 3.6 seconds for SIS and about 95 seconds for MS. However, the obtained MS and SIS estimators were equal to 1.761×10^{12} and 3.074×10^{11}, respectively, while the exact total number of independent sets in $\mathcal{A}_{35}$ (1), is equal to 1.787×10^{12}. That is, SIS is underestimating the true quantity due to its larger variance. Consequentially, the PMLB bounds of MS are tighter as compared to the ones provided by the SIS method. We also tried to run the SIS algorithm for a comparable running time, and executed it with 95 seconds timeout. In this case, the SIS estimator is equal to 6.271×10^{11}, which is closer to the analytical value. The exact RER of MS and SIS when reaching the 3% RE, is 1.46% and 82.8%, respectively. In contrast, the 95 seconds SIS run decreased



(a) The MS algorithm.

(b) The SIS algorithm.

Figure 2: The total number of independent sets, the PMLB which is equal to MSB for this problem, and the RE, as a function of the independent iteration number R, obtained for the $\mathcal{A}_{35}$ graph with the MS algorithm 2(a) using $N = 1000$ and $\rho = 0.2$, and the SIS algorithm 2(b), respectively. The dashed line stands for the analytical counting value of 1.787×10^{12}. The RE threshold for both algorithms is 3%.

the RER to 64.9%. We conjecture that the inferior performance of SIS is due to a special structure of Andrásfai graph, which is hard to approximate by the SIS relaxation, and thus the importance probabilities calculated by the SIS's DP are not close to the real ones.

4.2 Solving the #k-IS problem

Figure 3 summarizes the results for estimating #k-IS' of the $\mathcal{A}_{35}$ graph using the MS method. In particular, it shows the entire distribution of independent sets for different cardinalities. For each k, we run the MS algorithm until the predefined RE threshold of 3% is reached. The overall CPU time used by the MS algorithm is 15676 seconds. We examined the RER of the estimators via the comparison with the coefficients of the $\mathcal{A}_{35}$'s IP, and found that it did not exceed 6% for any $1 \leq k \leq 35$.

Solving the max-IS problem

From the IP in (1), we learn that $\alpha(\mathcal{A}_{35}) = 35$. Even though we managed to estimate the number of independent sets of cardinality 35 while calculating the distribution in Figure 3, it is of interest from the computational point of view to apply the binary search Algorithm 5 for estimating $\hat{\alpha}(\mathcal{A}_{35})$. Table 1

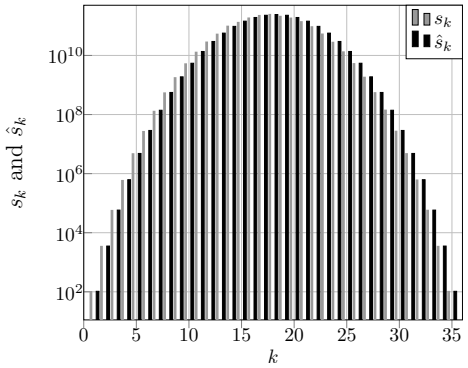


Figure 3: The analytical (s_k) and the estimated ($\hat{s}_k$) values of the k-cardinality independent sets in the $\mathcal{A}_{35}$ graph as a function of the set size k. The MS algorithm parameters are $N = 1000$ and $\rho = 0.2$. The RE threshold is set to 3%.

summarizes the performance of Algorithm 5 for the $\mathcal{A}_{35}$ graph. Note that $\hat{\alpha}(\mathcal{A}_{35}) = \alpha(\mathcal{A}_{35}) = 35$, and that it is much faster to find than the entire distribution in Figure 3.

Table 1: Performance of the MS Algorithm 5 on the $\mathcal{A}_{35}$ graph using $N = 1000$ and $\rho = 0.2$. The "# bin." stands for the number of binary search calls to the MS algorithm.

G	$\alpha(G)$	$\hat{\alpha}(G)$	# bin.	CPU (s)
$\mathcal{A}_{35}$	35	35	7	109.2

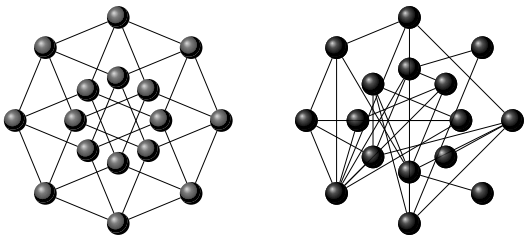
Note that, for the max-IS problem, we are not interested in an unbiased estimator, and consequently there is no need to obtain the fixed γ_t levels for $t = 1, \ldots, T$. That is, we can save the computational effort by using a pilot run of the MS algorithm 2 in line 4 of Algorithm 5.

While the $\mathcal{A}_{35}$ example is interesting from a benchmarking point of view, we next proceed with less trivial and larger examples to further examine the performance of the MS algorithm.

4.3 The Hypercube and the Waxman graphs

In this section we consider two families of graphs, the Hypercube [28] and the Waxman [54] graph. This example demonstrates the ability of the MS Algorithm 2 to handle networks with several hundreds of vertices. A brief description of these graph families is given below.

The d-Hypercube graph ($\mathcal{H}_d$) is a regular graph with 2^d vertices and $d\,2^{d-1}$ edges. In order to construct the Hypercube graph, label the 2^d vertices with



(a) The $\mathcal{H}_4$ graph. (b) The $\mathcal{W}_{16}$ graph.

Figure 4: An example of the Hypercube ($\mathcal{H}_4$) and Waxman ($\mathcal{W}_{16}(0.91, 0.25)$) graphs.

d-bit binary numbers, and connect every two vertices whenever the Hamming distance between their labels is 1. An example $\mathcal{H}_4$ graph is shown in Figure 4(a).

The Waxman graph ($\mathcal{W}_n(q, s)$) is a *spatially embedded random graph*, which is used for modelling of real-world graphs [4]. This graph is constructed as follows. Place n vertices in Euclidean space and connect every two vertices, u and v, with probability $q \exp\{-s \|u - v\|\}$, where $\|u - v\|$ is the Euclidean distance between u and v. An example $\mathcal{W}_{16}$ graph is shown in Figure 4(b).

Throughout this section we work with six different Hypercubes and Waxman graphs. Table 2 summarizes the graphs with their parameters.

Table 2: Summary of the graphs that are examined in Section 4.3: the Hypercubes and the corresponding Waxman graphs with the same number of vertices and edges.

$\mathcal{H}_d$	$\mathcal{W}_n(q, s)$	$\|V\|$	$\|E\|$
$\mathcal{H}_4$	$\mathcal{W}_{16}(0.91, 0.25)$	16	32
$\mathcal{H}_5$	$\mathcal{W}_{32}(0.55, 0.25)$	32	80
$\mathcal{H}_6$	$\mathcal{W}_{64}(0.29, 0.25)$	64	192
$\mathcal{H}_7$	$\mathcal{W}_{128}(0.18, 0.25)$	128	448
$\mathcal{H}_8$	$\mathcal{W}_{256}(0.09, 0.25)$	256	1024
$\mathcal{H}_9$	$\mathcal{W}_{512}(0.05, 0.25)$	512	2304

To ensure a fair comparison, the random Waxman graphs are generated subject to the constraint that they have the same number of vertices and edges as the corresponding Hypercube graphs.

Solving the #all-IS problem

We run both the MS and the SIS algorithms on all 12 graphs. Since we do not have analytical values for the Hypercube and the Waxman models, the RER is not available. However, for the #all-IS problem there exists an asymptotic result

for the Hypercube graphs [36]. In particular, for a d-dimensional Hypercube $\mathcal{H}_d$:

$$\lim_{d\to\infty} \frac{\#\text{all-IS}}{2^{2^{d-1}}} = 2\sqrt{e}.$$

We use the above equation to report Δ, which stands for the relative distance of the counting estimator to the asymptotic value. Namely,

$$\Delta = \frac{\left|2\sqrt{e}\ 2^{2^{d-1}} - \text{ counting estimator of MS/SIS}\right|}{2\sqrt{e}\ 2^{2^{d-1}}}.$$

Table 3: The total number of independent sets for the Hypercubes and the Waxman graphs with the MS algorithm using $N = 1000$ and $\rho = 0.2$, and the SIS algorithm, respectively. The RE threshold for both algorithms is 3%.

	MS				SIS			
G	R	counting estimator	Δ	CPU (s)	R	counting estimator	Δ	CPU (s)
$\mathcal{H}_4$	11	716	1.52×10^{-1}	0.13	50	753	1.08×10^{-1}	1.20×10^{-4}
$\mathcal{W}_{16}$	11	863	-	0.15	10	847	-	1.00×10^{-4}
$\mathcal{H}_5$	25	2.55×10^{5}	1.81×10^{-1}	0.98	223	2.58×10^{5}	1.94×10^{-1}	0.097
$\mathcal{W}_{32}$	20	3.07×10^{5}	-	0.76	42	2.91×10^{5}	-	0.02
$\mathcal{H}_6$	35	2.01×10^{10}	4.16×10^{-1}	4.89	1306	2.04×10^{10}	4.39×10^{-1}	2.77
$\mathcal{W}_{64}$	50	2.48×10^{10}	-	6.84	51	2.55×10^{10}	-	0.09
$\mathcal{H}_7$	143	7.90×10^{19}	2.99×10^{-1}	81.1	9744	7.73×10^{19}	2.71×10^{-1}	92.4
$\mathcal{W}_{128}$	117	5.43×10^{19}	-	65.8	166	5.96×10^{19}	-	1.65
$\mathcal{H}_8$	296	1.22×10^{39}	9.11×10^{-2}	652	1.19×10^{5}	1.26×10^{39}	1.22×10^{-1}	7081
$\mathcal{W}_{256}$	290	1.62×10^{38}	-	655	435	1.65×10^{38}	-	24.7
$\mathcal{H}_9$	638	4.21×10^{77}	1.03×10^{-1}	6109	8.05×10^{5}	4.65×10^{77}	2.18×10^{-1}	4.12×10^{5}
$\mathcal{W}_{512}$	687	1.21×10^{72}	-	6195	1005	1.20×10^{72}	-	542

Table 3 summarizes the performance of the MS and the SIS algorithms for the #all-IS problem. The MS and SIS counting estimators are in close proximity to one another for all instances, and to the asymptotic value for the Hypercubes. We can clearly observe that for the random Waxman graphs the SIS algorithm is much faster. However, the (structured) Hypercubes appear to be more challenging for the SIS method.

We conjecture that, similar to the $\mathcal{A}_{35}$ graph problem, this is due to the fact that the SIS relaxation is not adequate for such structured graphs. For example, when considering the $\mathcal{H}_9$ model, the SIS algorithm did not converge to the predefined 3% RE threshold in a reasonable time. In particular, while the MS algorithm converged to 3% RE in less than 2 hours, we stopped the SIS algorithm execution after 4 days, but its RE was still about 8%. The corresponding behavior of MS and SIS for the $\mathcal{H}_9$ model is summarized in Figure 5.

Solving the #k-IS problem

While acquiring the 3% RE estimators for large graphs is unmanageable in the sense of computation effort, we can still get fairly good bounds using the

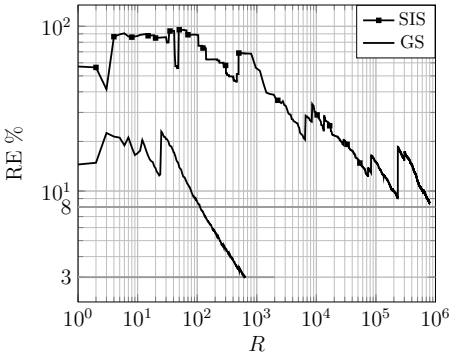


Figure 5: The RE as a function of the independent iteration number R of MS and SIS for the $\mathcal{H}_9$ model. MS and SIS converge to 3% and 8% RE, respectively.

PMLB. Our experiments indicate that the PMLB bound is tight, and it is much cheaper to obtain. The top and the bottom rows of Figure 6 summarize the 3% RE threshold and the corresponding PMLB results for estimating k-cardinality independent sets for the Hypercube and the Waxman graphs with $16, 32, 64$ and 128 vertices.

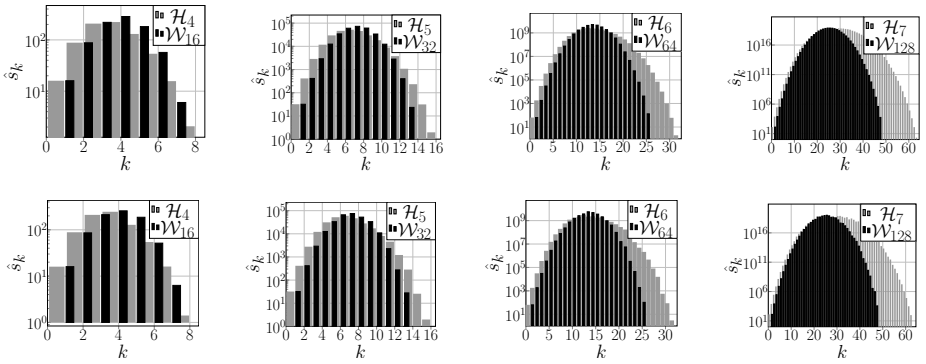


Figure 6: Distribution of the number of independent sets in Hypercubes and Waxman graphs using the MS algorithm with $N = 1000$ and $\rho = 0.2$. The RE threshold is set to 3%, and the PMLB=MASB (for $\omega = 0.95$) was constructed using ten independent runs of the MS algorithm.

Encouraged by the fact that the PMLB is very close to the 3% estimators in Figure 6, and since it is expensive to obtain the 3% RE estimators for the graphs with 256 and 512 vertices (the overall CPU times for full distribution calculation are given in Figure 8), we do it using PMLB only. Figure 7 summarizes the results for estimating k-cardinality independent sets for the Hypercube and the Waxman graphs with 256 and 512 vertices.

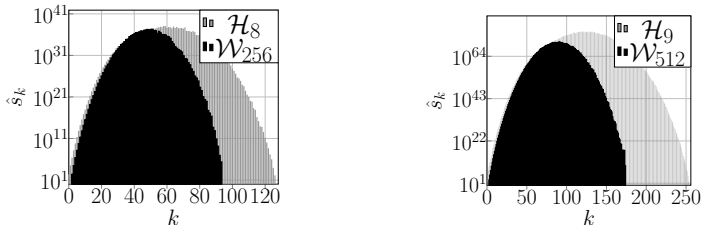


Figure 7: Distribution of the number of independent sets in Hypercubes and Waxman graphs using the MS algorithm with $N = 1000$ and $\rho = 0.2$. The PMLB=MASB (for $\omega = 0.95$) was constructed using ten independent runs of the MS algorithm.

Figures 6 and 7 indicate that the Hypercube graphs have a larger independence number. The latter can be useful in social network research, since this might give a helpful indication of the clique distribution in structured versus non-structured graphs such as Hypercubes and Waxman graphs.

Solving the max-IS problem

We do not know the independence number of Waxman random graphs. However, for Hypercubes it is known [56] that $\alpha(\mathcal{H}_d) = 2^{d-1}$. Table 4 summarizes the performance of the binary search Algorithm 5 applied to the Hypercube and the Waxman models. The results indicate that for these cases $(\mathcal{H}_4, \ldots, \mathcal{H}_9)$ the MS algorithm is always able to correctly identify the independence number of the Hypercube graphs.

Table 4: Performance of Algorithm 5 on the Hypercube and Waxman graphs using $N = 1000$ and $\rho = 0.2$.

G	$\alpha(G)$	$\hat{\alpha}(G)$	# bin.	CPU (s)
$\mathcal{H}_4$	8	8	4	0.22
$\mathcal{W}_{16}$	–	7	4	0.26
$\mathcal{H}_5$	16	16	5	1.38
$\mathcal{W}_{32}$	–	13	5	1.61
$\mathcal{H}_6$	32	32	6	13.5
$\mathcal{W}_{64}$	–	25	6	11.5
$\mathcal{H}_7$	64	64	7	104
$\mathcal{W}_{128}$	–	47	7	79
$\mathcal{H}_8$	128	128	8	957
$\mathcal{W}_{256}$	–	92	8	753
$\mathcal{H}_9$	256	256	9	9199
$\mathcal{W}_{512}$	–	172	9	6998

Next, we investigate the performance of Algorithm 5 for several non-trivial benchmarks.

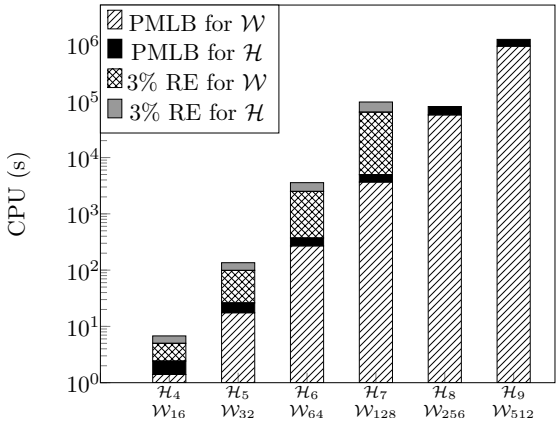


Figure 8: The summary of CPU times that are needed to acquire the full independent set distribution via the MS algorithm using $N = 1000$ and $\rho = 0.2$ for 3% RE threshold and the PMLB estimator, which is based on ten independent runs. For the $\mathcal{H}_8, \mathcal{H}_9, \mathcal{W}_{256}$ and $\mathcal{W}_{512}$ graphs, the calculation is performed with PMLB bounds only.

4.4 Determining the independence number for non-trivial graphs

In this section we explore four benchmarks from `https://turing.cs.hbg.psu.edu/txn131/clique.html`. The corresponding number of vertices, edges, and the performance of Algorithm 5 applied to these graphs are summarized in Table 5. The results indicate that the MS algorithm manages to solve these problems correctly in a reasonable time.

Table 5: Performance of Algorithm 5 on the benchmark problems using the MS algorithm with $N = 1000$ and $\rho = 0.2$.

G	$\|V\|$	$\|E\|$	$\alpha(G)$	$\hat{\alpha}(G)$	# bin.	CPU (s)
`keller4`	171	9435	11	11	7	327
`gen200_p0.9_44`	200	17910	44	44	7	396
`gen200_p0.9_55`	200	17910	55	55	8	822
`C125.9`	125	6363	≥ 34	34	7	96.1

Next, we are interested in checking if the independence number of `C125.9` is indeed 34. This is an unsolved problem. However, we can use the following idea to investigate the conjecture that $\alpha(\texttt{C125.9}) = 34$. While the proposed procedure certainly does not provide a formal proof, it can give some confidence that the independence number is equal to 34. We do the following.

- Run the MS algorithm to count independent sets of cardinality $k = 34$ and obtain a counting estimate within some small RE.

- During the execution of MS, accumulate unique independent sets of size 34.
- Finally, try to extend these accumulated cardinality 34 independent sets with an additional vertex to obtain a 35-IS.

Using $N = 10000$ and $\rho = 0.2$, we run the MS Algorithm 2, until reaching RE $= 0.5\%$ threshold. The resulting CPU time is 248102 seconds, and the number of independent splitting iterations is $R = 2233$. While running the MS algorithm, we also maintained the number of unique independent sets of cardinality 34.

We managed to obtain 924 unique independent sets of size 34, but none could be extended with an additional vertex to form a 35-IS. The counting estimator delivers a value of 914.4, and we conjecture that we found all independent sets of size 34, that is, $\alpha(\mathsf{C125.9}) = 34$. Figure 9 summarizes the convergence of MS. Out of 2.234×10^7 generated independent sets, the distribution of unique sets is almost uniform over the 924 solutions.

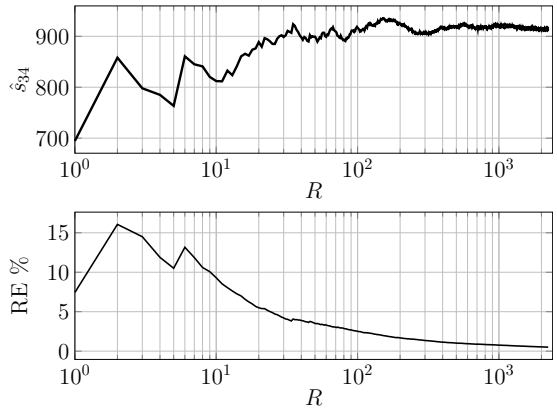


Figure 9: The total number of independent sets of cardinality 34 and the corresponding RE, as a function of the independent iteration number R, obtained for the C125.9 graph via the MS algorithm using $N = 10000$ and $\rho = 0.2$. The RE threshold is 0.5%.

We finish our numerical study by examining a simple graph, that serves as a warning to a MS algorithm user. In particular, we investigate a book graph [22].

4.5 The book graph

We consider the n-book graph $\mathcal{B}_n$. An example 6-book graph is given in Figure 10.

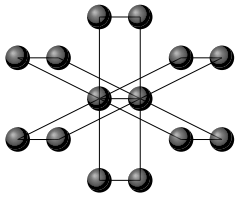


Figure 10: The 6-book graph, $\mathcal{B}_6$.

The $\mathcal{B}_n$ graph is defined as the Cartesian graph product $\mathcal{S}_{(n+1)} \times \mathcal{P}_2$, where $\mathcal{S}_{(n+1)}$ is a star graph and $\mathcal{P}_2$ is the path graph on two nodes [22]. The $\mathcal{B}_n$ IP is given by $2x(1+x)^n + (1+2x)^n$ [41].

Using our regular MS parameter set, $N = 1000$ and $\rho = 0.2$, we executed the distribution estimation experiment for the target RE threshold of 3%. Figure 11 summarizes the results. The exact RER is always less than 6% and the total CPU time for this experiment is 6910 seconds.

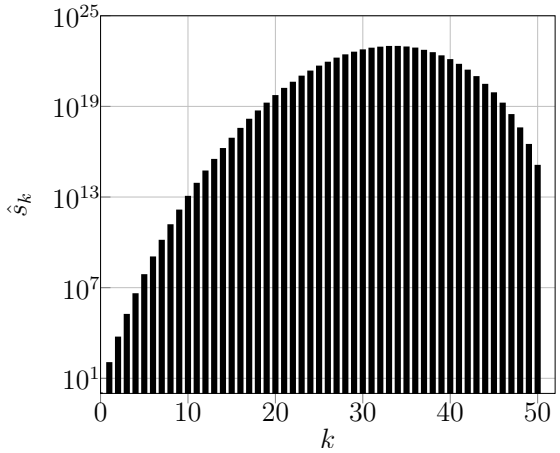


Figure 11: Distribution of independent sets in $\mathcal{B}_{50}$ graph using the MS algorithm with $N = 1000$ and $\rho = 0.2$. The RE threshold is 3%.

Our experiment managed to obtain an estimator for the number of all independent sets of cardinality k, for $1 \leq k \leq 50$. However, from the $\mathcal{B}_{50}$'s IP we learn that the independence number is equal to 51, that is, there exists at least one 51-IS that was not found by MS! Moreover, we tried to run the binary search Algorithm 5 using both $N = 10000$ and $N = 100000$ sample sizes, but the MS algorithm always failed to reveal a 51-IS.

The reason for this failure is as follows. Using the IP of the book graph, we calculated the exact number of 50-cardinality and 51-cardinality independent sets and got 1.1259×10^{15} and 2 independent sets of cardinalities 50 and 51, respectively. Recall the definition of our performance function S, which determines the level-sets; $S(\mathbf{x})$ is defined to be *the number of adjacent vertices* in $\mathbf{x}$.

Consequentially, the MS will reach the final level for which $S(\cdot) = 0$ with very small probability, which is equal to $2/1.1259 \times 10^{15}$. This is due to the fact that only two out of 1.1259×10^{15} 50-cardinality independent sets can be extended to 51-IS, since the $\mathcal{X}_{T-1}$ level set is much larger than the last one — $\mathcal{X}_T$. Indeed, comparing Figure 11 with Figures 3, 6 and 7, we can observe the much smoother behavior of the latter ones. That is, using the MS performance function defined in this paper, one should be careful, since only relatively smooth instances can be handled reliably.

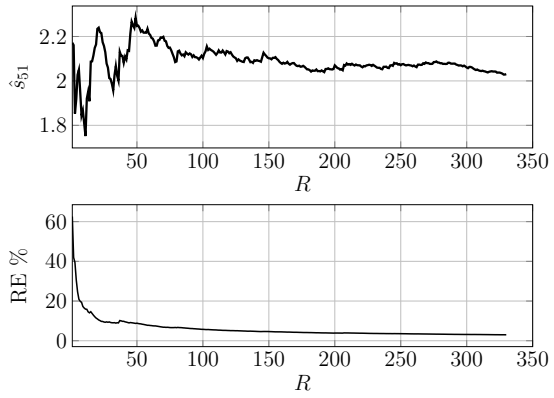


Figure 12: The total number of independent sets of cardinality 51 and the corresponding RE, as a function of the independent iteration number R, obtained for the $\mathcal{B}_{50}$ graph via the MS algorithm using $N = 1000$ and $\rho = 0.2$. The RE threshold is 3%.

In order to fix this problem, we need to resort to the $\mathcal{B}_n$ book graph topology. Denote the book central vertices by c and c', and let $U = \{u_{1,1}, \ldots, u_{1,n}\}$ and $U' = \{u'_{1,1}, \ldots, u'_{1,n}\}$ be the sets of adjacent vertices to c and c', respectively. Then, $\mathcal{B}_n$ has only two possible maximum independent sets, namely $\{c \cup U'\}$ and $\{c' \cup U\}$. Recall that when using the standard performance function, which counts the number of adjacent vertices in a set, the MS algorithm will generally fail to reach these favorable configurations. This can be repaired by introducing a bigger penalty for taking both u_i and u'_i (for any $i = 1, \ldots, n$), when they are adjacent. In our experiment, this penalty is equal to the number of graph edges. Such a performance function will cause MS to prefer taking a central vertex instead of an (u_i, u'_i) pair, and the MS algorithm converges. For example, using the new performance function the binary MS Algorithm 5 finds the max-IS in $\mathcal{B}_{50}$ after seven iterations.

A typical convergence pattern of the MS algorithm for counting of independent sets of cardinality 51 in the $\mathcal{B}_{50}$ graph is summarized in Figure 12. Note that we obtain $\hat{s}_{51} \approx 2.03$, as expected.

5 Conclusion

In this paper we introduced the Multilevel Splitting method for estimating the number of independent sets and the independence number in a graph. We constructed the corresponding Gibbs samplers, analyzed their performance in the sense of computational effort, and developed freely available software packages. The proposed algorithms are not very hard to implement, and our numerical study indicates that the practical performance is comparable with and sometimes better than currently existing state-of-the-art algorithms. In addition, applying the probabilistic lower bounds from Section 3.4 allows the study of the distribution of independent sets in graphs with several hundreds of vertices.

As for future work, we propose the following directions.

1. Specify graph topologies for which rigorous performance guarantees can be obtained.

2. Develop different performance functions that can be used for non-smooth graph instances. While it is definitely a challenging problem, a possible research direction is to use the smoothed splitting version of [13].

3. From a practical point of view, it will be of interest to develop a software package that runs on multiple CPUs or a GPU. This will allow us to handle larger graphs, and thus have significant impact on the field of graph analysis under the big-data setting.

Acknowledgements

We are thoroughly grateful to the anonymous reviewers and the associate editor for their valuable and constructive remarks and suggestions. In addition, we would like to thank the journal editor, Prof. Douglas R. Shier, for his priceless comments on how to improve the paper. This work was supported by the Australian Research Council Centre of Excellence for Mathematical & Statistical Frontiers, under CE140100049 grant number.

References

[1] R.D. Alba, *A graph-theoretic definition of a sociometric clique*, The Journal of Mathematical Sociology **3** (1973), no. 1, 113–126.

[2] B. Andrásfai, *Introductory graph theory*, Pergamon Press, Detroit, 1977.

[3] S. Asmussen and P.W. Glynn, *Stochastic simulation: Algorithms and analysis*, Applications of Mathematics, Springer, New York, 2007.

[4] L. Barnett, E. Di Paolo, and S. Bullock, *Spatially embedded random networks*, Phys. Rev. E **76** (2007), 056–115.

[5] C. Bazgan, B. Escoffier, and V.T. Paschos, *Completeness in standard and differential approximation classes: Poly-(d)apx- and (d)ptas-completeness*, Theoretical Computer Science **339** (2005), no. 23, 272–292.

[6] J. Blanchet and D. Rudoy, *Rare event simulation and counting problems*, Rare Event Simulation Using Monte Carlo Methods (G. Rubino and B. Tuffin, eds.), John Wiley & Sons, London, 2009, pp. 171–193.

[7] J.K. Blitzstein and P. Diaconis, *A sequential importance sampling algorithm for generating random graphs with prescribed degrees*, Internet Mathematics **6** (2011), no. 4, 489–522.

[8] Z.I. Botev and D.P. Kroese, *Efficient Monte Carlo simulation via the generalized splitting method*, Statistics and Computing **22** (2012), 1–16.

[9] N. Bourgeois, B. Escoffier, V.T. Paschos, and J.M.M. Rooij, *A bottom-up method and fast algorithms for max independent set*, Proceedings of Algorithm Theory - SWAT 2010: 12th Scandinavian Symposium and Workshops on Algorithm Theory, 2010, pp. 62–73.

[10] R.E. Bryant, S.M. German, and M.N. Velev, *Microprocessor verification using efficient decision procedures for a logic of equality with uninterpreted functions*, Proceedings of the International Conference on Automated Reasoning with Analytic Tableaux and Related Methods (London, UK), Springer-Verlag, 1999, pp. 1–13.

[11] G. Casella and E.I. George, *Explaining the Gibbs sampler*, The American Statistician **46** (1992), no. 3, 167–174.

[12] F. Cérou, P. Del Moral, T. Furon, and A. Guyader, *Sequential Monte Carlo for rare event estimation*, Statistics and Computing **22** (2012), no. 3, 795–808.

[13] F. Cérou, A. Guyader, R.Y. Rubinstein, and R. Vaisman, *On the use of smoothing to improve the performance of the splitting method*, Stochastic Models **27** (2011), no. 4, 629–650.

[14] Y. Chen, P. Diaconis, S.P. Holmes, and J.S. Liu, *Sequential Monte Carlo methods for statistical analysis of tables*, Journal of the American Statistical Association **100** (2005), 109–120.

[15] S.E. Chick, *Sequential sampling with economics of selection procedures*, Management Science **58** (2012), no. 3, 550–569.

[16] S.A. Cook, *The complexity of theorem-proving procedures*, Proceedings of the Third Annual ACM Symposium on Theory of Computing (New York, NY, USA), STOC '71, ACM, 1971, pp. 151–158.

[17] P.T. de Boer, D.P. Kroese, S. Mannor, and R.Y. Rubinstein, *A tutorial on the cross-entropy method*, Annals of Operations Research **134** (2002), 19–67.

[18] T. Dean and P. Dupuis, *Splitting for rare event simulation: A large deviation approach to design and analysis*, Stochastic Processes and their Applications **119** (2009), no. 2, 562 – 587.

[19] T. Dean and P. Dupuis, *The design and analysis of a generalized RESTART/DPR algorithm for rare event simulation.*, Annals of Operations Research **189** (2011), 63–102.

[20] P. Del Moral, *Feynman-kac formulae: Genealogical and interacting particle systems with applications*, Probability and Its Applications, Springer, New York, USA, 2004.

[21] P. Doreian and K.L. Woodard, *Defining and locating cores and boundaries of social networks*, Social Networks **16** (1994), no. 4, 267–293.

[22] J.A. Gallian, *A dynamic survey of graph labeling*, Electronic J. Combinatorics **16** (2008), 1–58.

[23] P. Glasserman, P. Heidelberger, P. Shahabuddin, and T. Zajic, *Multilevel splitting for estimating rare event probabilities*, Operations Research **47** (1999), no. 4, 585–600.

[24] C. Godsil and G. Royle, *Algebraic graph theory*, Graduate Texts in Mathematics, vol. 207, Springer, New York, 2001.

[25] V. Gogate and R. Dechter, *A new algorithm for sampling CSP solutions uniformly at random*, pp. 711–715, Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.

[26] V. Gogate and R. Dechter, *Approximate counting by sampling the backtrack-free search space*, Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence, July 22-26, 2007, Vancouver, British Columbia, Canada, AAAI Press, 2007, pp. 198–203.

[27] V. Gogate and R. Dechter, *Sampling-based lower bounds for counting queries*, Intelligenza Artificiale **5** (2011), no. 2, 171–188.

[28] F. Harary, J.P. Hayes, and H. Wu., *A survey of the theory of hypercube graphs*, Computers & Mathematics with Applications **15** (1988), no. 4, 277–289.

[29] C. Hoede and X. Li, *Clique polynomials and independent set polynomials of graphs*, Discrete Mathematics **125** (1994), no. 1, 219–228.

[30] M. Jerrum and A. Sinclair, *The Markov chain Monte Carlo method: an approach to approximate counting and integration*, Approximation Algorithms for NP-hard Problems (Boston) (D. Hochbaum, ed.), PWS Publishing, 1996, pp. 482–520.

[31] H. Kahn and T.E. Harris, *Estimation of particle transmission by random sampling*, National Bureau of Standards Applied Mathematics Series **12** (1951), 27–30.

[32] R.M. Karp, *Reducibility among combinatorial problems*, Complexity of Computer Computations (R.E. Miller and J.W. Thatcher, eds.), Plenum, New York, 1972, pp. 85–103.

[33] R.M. Karp and M. Luby, *Monte-Carlo algorithms for enumeration and reliability problems*, Proceedings of the 24th Annual Symposium on Foundations of Computer Science, 1983, pp. 56–64.

[34] F.P. Kelly, *Stochastic models of computer communication systems*, Journal of the Royal Statistical Society. Series B (Methodological) **47** (1985), no. 3, 379–395.

[35] D.E. Knuth, *Estimating the efficiency of backtrack programs*, Math. Comp. **29** (1975), 122–136.

[36] A.D. Korshunov and A.A. Sapozhenko, *The number of binary codes with distance 2*, Problemy Kibernet. **40** (1983), 111–130.

[37] D. Kroening and O. Strichman, *Decision procedures: An algorithmic point of view*, Springer, New York, 2008.

[38] D.P. Kroese, T. Taimre, and Z.I. Botev, *Handbook of Monte Carlo methods*, John Wiley & Sons, New York, 2011.

[39] L. Lavagno, G. Martin, and L. Scheffer, *Electronic design automation for integrated circuits handbook - 2 volume set*, CRC Press, Inc., Boca Raton, FL, USA, 2006.

[40] J.V. Leeuwen, *Handbook of theoretical computer science: Algorithms and complexity*, MIT Press, Cambridge, MA, 1990.

[41] V.E. Levit and E. Mandrescu, *The independence polynomial of a graph – a survey*, Proceedings of the First International Conference on Algebraic Informatics, Aristotle Univ., 2005, pp. 233–254.

[42] D. Nau, M. Ghallab, and P. Traverso, *Automated planning: Theory & practice*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2004.

[43] Gerardo Rubino and Bruno Tuffin (eds.), *Rare event simulation using Monte Carlo methods*, John Wiley & Sons, London, 2009.

[44] R.Y. Rubinstein, *The Gibbs cloner for combinatorial optimization, counting and sampling*, Methodology and Computing in Applied Probability **11** (2009), 491–549 (English).

[45] R.Y. Rubinstein, A. Dolgin, and R. Vaisman, *The splitting method for decision making*, Communications in Statistics - Simulation and Computation **41** (2012), no. 6, 905–921.

[46] R.Y. Rubinstein and D.P. Kroese, *Simulation and the Monte Carlo Method*, Second ed., John Wiley & Sons, New York, 2008.

[47] R.Y. Rubinstein, A. Ridder, and R. Vaisman, *Fast sequential Monte Carlo methods for counting and optimization*, John Wiley & Sons, New York, 2013.

[48] T. Sang, F. Bacchus, P. Beame, H. Kautz, and T. Pitassi, *Combining component caching and clause learning for effective model counting*, Seventh International Conference on Theory and Applications of Satisfiability Testing, 2004, pp. 1–16.

[49] A. Sly, *Computational transition at the uniqueness threshold*, 2010 IEEE 51st Annual Symposium on Foundations of Computer Science, 2010, pp. 287–296.

[50] O. Strichman, *Efficient decision procedures for validation: Translation validation, decision procedures for equality logic, and sat tuning for bounded model checking*, Lambert Acad. Publ., Saarbrücken, 2009.

[51] S.P. Vadhan, *The complexity of counting in sparse, regular, and planar graphs*, SIAM Journal on Computing **31** (1997), 398–427.

[52] R. Vaisman, Z.I. Botev, and A. Ridder, *Sequential Monte Carlo for counting vertex covers in general graphs*, Statistics and Computing (2015), 1–17.

[53] L.G. Valiant, *The complexity of enumeration and reliability problems*, SIAM Journal on Computing **8** (1979), no. 3, 410–421.

[54] B.M. Waxman, *Routing of multipoint connections*, IEEE Journal on Selected Areas in Communications **6** (1988), no. 9, 1617–1622.

[55] D. Weitz, *Counting independent sets up to the tree threshold*, STOC '06: Proceedings of the Thirty-eighth Annual ACM symposium on Theory of Computing (New York, NY, USA), ACM, 2006, pp. 140–149.

[56] D.B. West, *Introduction to graph theory*, Second ed., Prentice Hall, New Jersey, September 2000.