Community Detection Through Polynomials

Neophytos Charalambides, Anthony Della Pella

neochara@umich.edu, adellape@umich.edu

December 8th, 2017

1 Abstract

A study was carried out comparing two distinct algorithms which approximate two different partition functions for a given model – one determined by certain combinatorial properties of the underlying graph, and the other by a graphical model. In the first case [4] a recently developed "approximation by interopolation" method is used, while in the second [9] the approximation is done in order to compute a constant factor approximation of a general integral over an exponentially large set.

While it was initially proposed that the two partition functions were equivalent, we soon found that this was not in fact true. It is possible though under certain models to achieve the sought equivalence.

A novel application of the first partition function we mention to community detection was also considered. This application uses the well known Erdős-Rényi graph to empirically obtain a well known result regarding the planted clique. Implementation of both approximations took place, where computation was a major factor due the nature of the partition functions. It was shown the algorithm in [9] does not work for relatively small random graphs, something we later found to be stated in [1].

2 Introduction

2.1 What is a partition function?

From Barvinok in [3]: "The answer depends on who you ask. You get one (multi)set of answers if you ask physicists, and another (multi)set if you ask mathematicians (we allow multisets, in case we want to account for the popularity of each answer)". We will adopt a combinatorial view of partition functions for our purposes, but their applications to physics need not be understated. In particular, it was shown in the seminal work of Heilman Lieb [12] that the Monomer-Dimer model from physics exhibits no phase transitions. Their proof utilizes a partition function and via showing that the partition function has no singularities near the positive real axis conclude the result.

Given a family $\mathcal{F}$, of subsets of the set $[n]$, we define the *partition function* of $\mathcal{F}$ as the polynomial in n real or complex variables $x_1, \dots, x_n$ as:

$$p_{\mathcal{F}}(x_1, \dots, x_n) = \sum_{S \in \mathcal{F}} \prod_{i \in S} x_i.$$

In general it is computationally difficult to compute exact values of partition functions for arbitrary choices of the x_1, however efficient *approximation* algorithms do exist for some partition functions such as the independence polynomial, the matching polynomial, the permanent, and the partition function of cliques described below.

2.2 Partition Function of Cliques

Throughout this report, we focus greatly on two specific partition functions. The first being the partition function of cliques [4] (defined for a given $n \times n$ matrix W and $1 < m \leq n$):

$$P_m(W) = \sum_{\substack{S \subseteq [n] \\ |S| = m}} \prod_{\substack{\{i,j\} \subseteq S \\ i \neq j}} w_{i,j}. \tag{1}$$

This function is interesting in the context of graph theory, as well as random graph theory for our purposes, in which we show a novel application. If known exactly for the adjacency matrix W of a graph G, then $P_m(W)$ is the number of cliques in the graph G of size m. Recall a clique of size m is a totally connected component or equivalently a copy of K_m, the complete graph on m vertices found within the graph G. Moreover, in [4] an algorithm is given which provided we can approximate $P_m(W)$ efficiently, allows us to efficiently and with high probability find highly connected collections of m vertices (*i.e.* an approximate clique).

2.3 WISH Algorithm

In [9] a similar partition function to that of the partition function of cliques is approximated utilizing a randomization procedure. The algorithm they use to approximate this is given the acronym WISH, for Weighted-Integrals-And-Sums-By-Hashing.

A prespecified graphical model is considered as a factor graph, with $N = |V|$ discrete random variables $x_i \in \mathcal{X}_i$ for $i \in V$ and $\mathcal{X} = \mathcal{X}_1 \times ... \times \mathcal{X}_N$. The factors taken into account are defined as $\psi_\alpha : \{x\}_\alpha \to \mathbb{R}_+$, for which α is taken from an index set $\mathcal{I}$ and $\{x\}_\alpha \subseteq V$.

For the model we focus on, we take $\mathcal{X}_i = \mathbb{Z}/2\mathbb{Z}$. Therefore $\mathcal{X} = \mathbb{Z}/2\mathbb{Z}^N$ is the set of all possible configurations. The weight function $w : \mathcal{X} \to \mathbb{R}_+$ we use is $w(x) = \prod_{\alpha\in\mathcal{I}} \psi_\alpha(\{x\}_\alpha)$, and the partition function in [9] is then defined by:

$$Z = \sum_{x\in\mathcal{X}} \prod_{\alpha\in\mathcal{I}} \psi_\alpha(\{x\}_\alpha). \tag{2}$$

Computing Z above is computationally expensive and intractable, as it involves a sum over an exponential number of configurations over the field of two elements. Below is the pseudocode of the algorithm, as presented in [9].

The inputs of the algorithm are the weight function w, the fixed constants δ and α which arise in the analysis of the algorithm with the only restriction that they are positive, though further work is shown to bound α from above by 0.00042, in order to relate it with δ in the MAP problem. As shown n is taken as input, and for our purposes we use it within WISH to construct the set of all configurations Σ.

2.4 Planted Clique Problem

The planted clique problem is often stated in terms of random Erdős-Rényi graphs as follows.

Algorithm 1 WISH ($w : \Sigma \to \mathbb{R}_+, n = log_2|\Sigma|, \delta, \alpha$)

```
$T \leftarrow$ round-up$\left(\frac{ln(1/\delta)}{\alpha} ln(n)\right)$
**for** $i = 0, ..., n$ **do**
    **for** $t = 1, ..., T$ **do**
        Sample $A \in \{0,1\}^{i\times n}$ and $b \in \{0,1\}^i$
        $w_i^t \leftarrow \max_\sigma w(\sigma)$ subject to $A\sigma \equiv b \mod 2$
        *Assume access to an optimization oracle above*
    **end for**
    $M_i \leftarrow$ Median$(w_i^1, ..., w_i^T)$
**end for**
return $Z \approx M_0 + \sum_{i=0}^{n-1} M_{i+1} 2^i$
```

Suppose $G \sim \mathcal{G}(n,p)$ is an Erdős-Rényi graph with vertex set $V = \{1, \dots, n\}$ and probability of edge connection $p \in [0,1]$. There are many combinatorial questions that have been asked regarding Erods Renyi random graphs, such as:

- when does G have a giant component?
- when is G connected?
- what is the largest clique inside G?

There are (at least partial) answers to these problems. More information is provided in the related work section below as well as the survey.

We can plant a clique on a random Erdős-Rényi graph with the following process. First we sample $G \sim \mathcal{G}(n,p)$, and then add a clique to a random subset of m vertices of G with $m \leq n$. Later, calling the class of graphs generated from this procedure $\mathcal{G}(n,p,m)$, we wish to find the vertices where the clique was planted given a graph $G \sim \mathcal{G}(n,p,m)$.

When $m \leq 2\log_2(n) + 1$ there is no hope for recovery, while for $m > 2\log_2(n) + 1$ there is a naive $n^{\mathcal{O}(\log n)}$ algorithm for finding the m desired vertices. An important question is then to determine the smallest value of m for which we have a polynomial time algorithm which can locate the vertices inducing the clique. From the plots below, we get a sense as to why (while an easy problem in low dimension n) finding such sub graphs is difficult.

2.5 Applications of the Partition Function of Cliques

The question above regarding the largest clique in a graph was one of the first applications found for the partition function of cliques. The authors utilize the algorithm in [4] in the results section on graphs with planted cliques in order to provide evidence that this function could lead to progress beyond the current state of the art. In particular, the partition

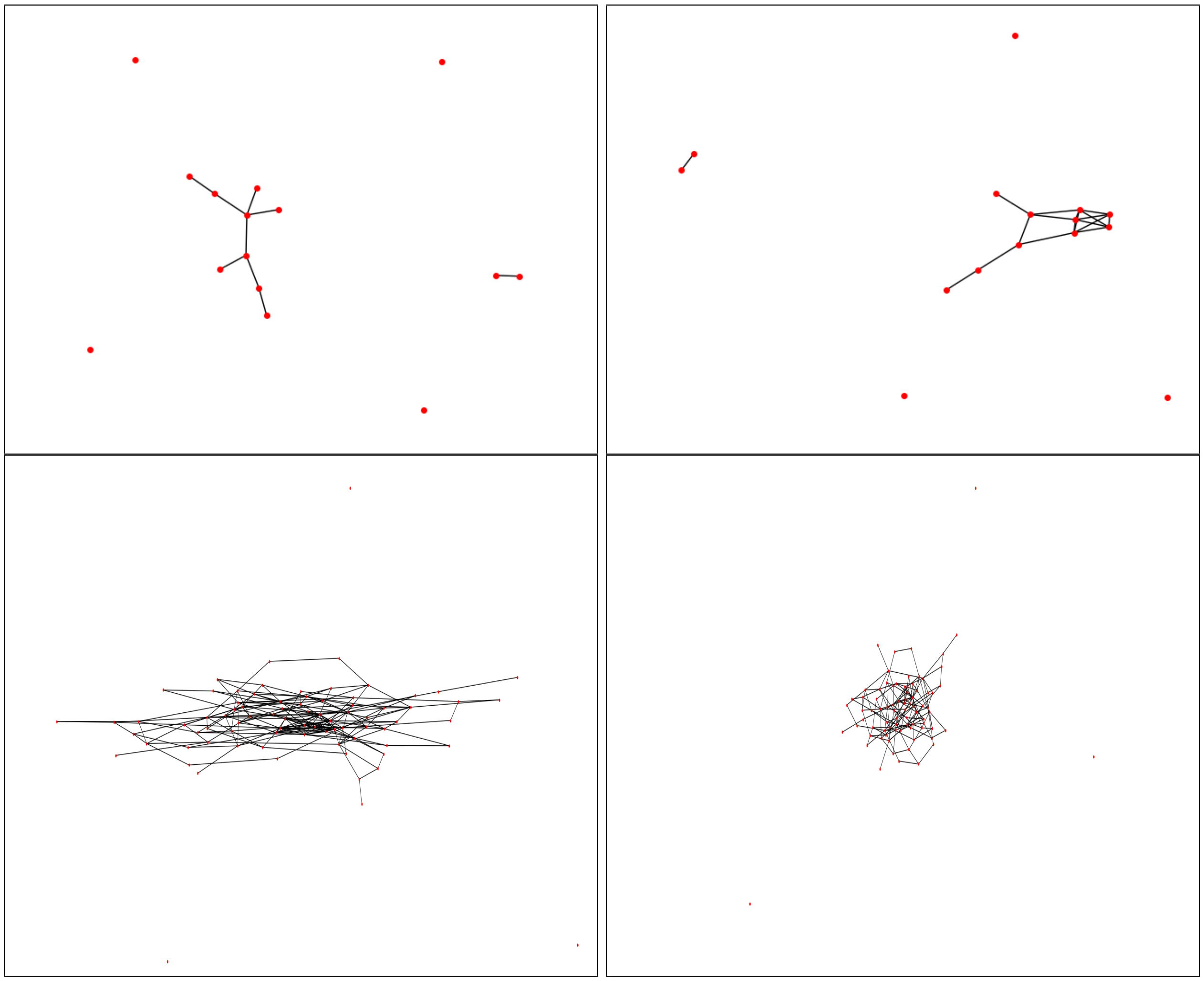


Fig. 1: On the top left is a regular graph $G \sim \mathcal{G}(15, .1)$ with no planted clique. To the right is G with a planted K_5. The bottom plot shows a graph $G \sim \mathcal{G}(80, 0.1)$ and then to the right is the same graph with a planted copy of K_9.

function P_m considered therein provides a way of calculating the number of cliques in a in a graph G. Thus, to find the largest clique it suffices to compute the function P_m on the adjacency matrix $W(G)$ for successive values of m. At the first such k where $P_m(W) = 0$, we conclude that the largest clique in G is $k - 1$.

Applying the partition function of cliques to this problem appears to be novel and utilizes the density functional described in the implementations section. In particular, there is a threshold given in [4] which given $\text{Density}_m(G)$, states that if $\text{Density}_m(G) \leq f(\sigma)$ then there are no subsets of G with density higher than σ. Using this result, the paper gives an algorithm which allows us to find a subset of m vertices in G with high density in polynomial time. Thus, we can detect and produce our clique on m vertices.

2.6 Clique Structured Ising Model

The Ising model is one of the most studied models nowadays, originating from the 1920's when it was invented by physicists Wilhelm Lenz and Ernst Ising. Our focus regarding the WISH algorithm was mainly on the Clique Structured Ising Model, for which not much literature was found. We chose to study this model mainly due to it lending itself most readily to the comparison with (2).

We therefore considered the clique structured Ising model on n binary variables $x_i \in \{0, 1\}$ for $i \in \{1, ..., n\}$, where the the weight between two variables w_{ij} is taken from a uniform distribution ranging from $[0, w\sqrt{|i - j|}]$, for which the authors of [9] set the parameter w to be constant at $w = 0.2$. The factors are then defined as $\psi_{ij}(x_i, x_j) = e^{-w_{ij}}$ when $x_i \neq x_j$ and $\psi_{ij}(x_i, x_j) = 1$ when $x_i = x_j$.

Then, given these factors a corresponding weighted adjacency matrix $\Psi = \left(\psi_{ij}(x_i, x_j)\right)_{i,j=1}^{n}$ is set up. This yields two cliques (which all edges have weight 1) for each binary string of length n (other than the strings consisting of only 0's or only 1's) which have factors of weight 1 between all variables of the same assignment (1 or 0).

2.7 Comparison of Algorithms

Initially we planned to tackle the two algorithms separately and then combine our work and implementations, compare the algorithms' performances and complexity. This comparison was to be done both through simulations and on real world data sets. It became clear that we would have to show that after some adaptations the one partition function would yield the other and vice versa.

This comparison procedure was found to be impossible in the general setting due to the non-equivalence of the partition functions. We further discuss this in section 5.

3 Related Work

3.1 Combinatorics and Partition Functions

There have been several recent bodies of work pertaining to partition functions in the same manner we are considering them, that is to say not motivated by theoretical physics.

Firstly, there have been connections to the computational complexity of these partition functions which is tied into the complex zeros of the corresponding function.

Secondly, there is a notion of phase transitions in physics which correlates (but not directly translates) to the domain of combinatorial partition functions as we study here.

The results regarding (1) owe themselves to the first body of work – by carefully examining the zeros of the partition function on the complex plane, the author was able to establish the algorithm using the interpolation method. This technique has been extended to find "good" algorithms for other partition functions as well such as the Permanent, the graph homomorphism partition function and the Hafnian. For further references see: [6],[7],[5], and for an overall introduction to the theory: [3].

3.2 Planted Clique Problem

The planted clique problem drew our interest later in the projects lifecycle – thus, the results are limited, but a fairly detailed account of the known results for the problem as well as future directions follows. Given a graph $G \sim \mathcal{G}(n, .5, m)$ (*i.e.* Erdős-Rényi with planted clique of size m on n vertices) a naive algorithm exists to answer the problem when $m > 2\log_2(n) + 1$. The idea behind the algorithm is to check all subsets S with $|S| = 2\log_2(n) + 2$ which induce a clique on G, then outputs $S \cup \{v\}$ for v a vertex with (v, w) an edge for all w in S. This procedure will produce a desired clique of size m. This algorithm doesn't run in polynomial time however and only accomplishes $n^{\mathcal{O}(\log(n))}$ time.

A natural question to ask at this point is what is the smallest m for which we have a polynomial time algorithm. The proof for $m \geq c(n \log n)^{1/2}$ relies on elementary techniques from probability such as the Hoeffding inequality and union bound. In [2] it was shown that for $m = cn^{1/2}$ such an algorithm exists. The problem overall has been resistant to a solution, and a possible reason for this was given in [10] with a follow up paper in 2015. This result is interesting as it extended the statistical query model to optimization problems.

3.3 Optimization Through Binary Codes

The idea of using a binary constraint, such as $A\sigma \equiv b \mod 2$ described in our report over randomly sampled hash functions A, has a purpose of reducing computation.

In [8], an improvement on the WISH algorithm proposed. The authors compute both lower and upper bounds on the partition function, which are tighter than those derived from other methods. For the improvement of WISH, the hash function A is not chosen totally in random from $\mathbb{Z}/2\mathbb{Z}^{i \times n}$, but rather to being a random *Toeplitz* matrix in the given space. The constraint considered is also reformulated to being a Linear Integer Program.

One of the issues we show in our work, how the sampling of the hash functions considered for WISH need be over large values of n. In [11] the authors propose a generalized importance sampling scheme based on the idea of randomly selecting exponentially large to obtain estimates of statistics, such as the partition function of undirected graphical models.

One of the more important papers following up on the WISH algorithm, is [1]. This paper helps better understand the performance of WISH and how to (partially) resolve some of its the major obstacles one faces when using it. It also reassures two of our findings and what made it difficulty to draw satisfactory results using the algorithm. By this we mean that XORs are computationally intractable with many variables, and have poor statistical performance with few variables. To go around these issues, they maximize the unnormalized probability function f, and solve systems of linear equations.

Further details on these can be found in our survey.

4 Proposed Methods

4.1 Initial Equivalence

At the onset of the project, we planned to show the equivalence of the two partition functions being described in the papers [4] and [9]. The goal of showing this equivalence was to provide us the ability to compare the two algorithms presented in the papers. We hoped to show superiority of one algorithm over the other empirically, and then compare this result to the theoretical bounds given in each reference.

If such a correspondence was shown it would be useful to forwarding progress on the community detection problem as a whole. Had the correspondence been shown, we would have used one of the current state of the art method of community detection (*i.e.* the spectral method [13]) to compare our results using the implementations of the algorithms below.

In particular, we planned to test on a real world data set, the Chicago taxi data. We had several issues in doing so, mainly due to the complexity of the data set and the fact that this data set consists primarily of relational data rather than the better suited associational data.

4.2 Implementation of Partition Function of Cliques Approximation Algorithm

The algorithm of Barvinok [4] for computing the clique partition function relies on three primary phases:

- Modification of adjacency matrix.
- Computation of k^{th} derivatives of $g(t)$.
- Recovery of Taylor approximation of $f(t)$ via linear equation solution.

Phase one of the algorithm sees a modification to the adjacency matrix W of our given graph. The particular modification depends on the size m of cliques that are desired for detection. In particular, for the adjacency matrix W of the graph, we change all instances of 0 (indicating there is no edge in between the corresponding vertices) to a parameter α. α is chosen so that $|w_{ij}-1| \leq \gamma/(m-1)$ where $\gamma > 0$ is an absolute constant. In [4], choosing $\gamma = .07$ suffices, while in the case of large graphs we can increase this to $\gamma = .27$ provided the size of cliques desired is sub linear in the number of vertices of the graph. A current area of research for the second author is determining the extent to which γ can be pushed to 1. See the discussion below regarding γ for more information on the theory behind why this change is necessary.

Phase two of the algorithm computes the derivatives $\frac{\mathrm{d}^j}{\mathrm{d}t^j} g(t)$ where $g(t)$ is the shifted clique partition function: $P_m(J + t(W - J))$ with J the matrix of all $1's$.

Phases and two are completed using the C++ language. Initially this was done in Python, however due to the computationally intensive nature of the problem a more optimized version of the code was written in C++ to improve results.

With access to these derivatives, the third phase of the algorithm computes a Taylor approximation to $f(t) = \ln g(t)$ at 1 where $f(1) = \ln P_m(W)$. The method consists of solving a triangular system of equations obtained from the derivatives $\frac{\mathrm{d}^1}{\mathrm{d}t^1}, \frac{\mathrm{d}^2}{\mathrm{d}t^2}, \ldots, \frac{\mathrm{d}^k}{\mathrm{d}t^k}$. Here, the order of the Taylor polynomial determines the quality of the approximation, however it suffices to choose k on the order of $\ln(m)$. This result in [4] is the main driving force behind the speed of the algorithm and is what gives the interpolation method it's power.

No numerical work has yet been done in computing P_m for specific graphs. It was one of the aims of this project to determine (using both simulated data and the data sets provided in the course) some exact instances of P_m to compare the algorithm above to. Unfortunately due to the nature of the data sets, this application never came to light. Simulated data was obtained from the Networkx python package however and the algorithm was run on these graphs. The theory states that by paying an $n^{\ln(\epsilon)}$ factor in computation one can calculate an ϵ approximation of P_m.

The procedure for generating random graphs consisted of using networkx's built in Erdős-Rényi graph method. Random Erdős-Rényi graphs were generated and their adjacency matrix ran through the algorithm. Then, the solution of the triangular system of equations was done in Mathematica with the built in equation solver. Code is provided in the supplement.

The table below represents data obtained from the algorithm. Each entry represents one generation of a graph on $n = 15$ vertices with vertex incidence probability corresponding to the p value in the first column. Going across rows determines the value of the clique that was being "tested". Formally, each value in the table is a degree 2 Taylor approximation of the quantity $\ln P_m(W)$ with W taken to be the modified adjacency matrix:

$$w_{ij} = \begin{cases} 1 + \dfrac{\gamma}{m-1} & \text{if } (i,j) \in E \\ 1 - \dfrac{\gamma}{m-1} & \text{if } (i,j) \notin E \end{cases}.$$

Here, γ was chosen to be $.45$ which agrees with results from the second author extending those in [4].

For a given subset S of vertices of a graph G, we let $\sigma(S)$ be the density (that is ratio of possible edges to appearing edges) between nodes in S. $\sigma(S)$ is always a quantity between 0 and 1 for simple graphs and is 0 precisely when S is an independent set. Suppose that we are given a graph G with n vertices and an integer $1 < m \leq n$. We can then define the density functional:

$$\text{Density}_m(G) = \sum_{\substack{S \subseteq V \\ |S|=m}} \exp\{\gamma m \sigma(S) - \epsilon(m)\}$$

$$\text{where } 0 \leq \epsilon(m) \leq \frac{.1}{m-1}$$

and γ is an absolute constant. We can view our partition function then as enumerating dense subsets except where we weigh down those sets with few inter connections being weighed down by a factor of $1-\gamma$ to a power corresponding to the number of missing connections. Also noted in [4] is the equality:

$$\text{Density}_m(G) = \exp\{\gamma m\} \left(1 + \frac{\gamma}{m-1}\right)^{-\binom{m}{2}} P_m(W).$$

This interpretation as well as the following facts explain why this density functional is a good indicator of the existence of cliques in a graph. Fix two real numbers $0 \leq \varsigma < \varsigma' \leq 1$. If there are no m subsets S with density ς or higher then

$$\text{Density}_m(G) \leq \binom{n}{m} \exp\{\gamma m \varsigma - \epsilon(m)\}.$$

On the other hand, if there are sufficiently many dense $m-$ subsets S with density ς' or higher, then

$$\text{Density}_m(G) \geq 2\binom{n}{m} \exp\{\gamma m \sigma - \epsilon(m)$$

Moreover, we can distinguish between these two cases with the above algorithm in $n^{\mathcal{O}(\ln \Updownarrow)}$ time. These inequalities also give rise to the procedure for producing an m-subset with

$$\exp\{\gamma m \sigma(S)\} \geq \frac{1}{2}\binom{n}{m}^{-1} \text{Density}_m(G).$$

Density of Erdős-Rényi			
p-values	m=3	m=5	m=7
p=0.05	7.296	9.357	9.761
p=0.10	7.303	9.403	9.872
p=0.15	7.300	9.450	9.835
p=0.20	7.310	9.498	10.093
p=0.25	7.313	9.529	10.204
p=0.35	7.315	9.359	10.333
p=0.35	7.326	9.601	10.370
p=0.40	7.326	9.581	10.481
p=0.45	7.326	9.568	10.647
p=0.50	7.341	9.687	10.536
p=0.55	7.345	9.740	10.776
p=0.60	7.345	9.727	10.795
p=0.65	7.349	9.792	10.850
p=0.70	7.353	9.766	11.016
p=0.75	7.355	9.865	11.053
p=0.80	7.359	9.865	11.219
p=0.85	7.364	9.938	11.404
p=0.90	7.372	9.984	11.386
p=0.95	7.372	10.003	11.533

4.3 Implementation of WISH algorithm

Our second algorithm of interest, the WISH algorithm, was implemented for the given clique structured Ising model described in section 2.6. One difficulty we faced was the optimization given the constraint $A\sigma \equiv \mod 2$.

In order to take into account all possible configurations of a fixed size, we construct a matrix $\Sigma \in \mathbb{Z}/2\mathbb{Z}^{2^N \times N}$ where each row is a distinct configuration. The implementation took place solely in matlab.

5 Results

5.1 Non Equivalence

From the description of the clique structured Ising model, along with (1) and (2), it first seemed as if the matrix Ψ was very similar to the weighted adjacency matrix defined in the process of the algorithm described in [4]. In this matrix the weights are assigned to $w_{ij} = 1$ if $\{i,j\} \in E$ and $w_{ij} = \alpha$ if

$\{i,j\} \notin E$ where α is a constant between 0 and 1, as described below in (7).

The overall picture is not that simple, as for (2) we consider all possible configurations. This can be viewed as "averaging" through all possible structures.

Looking at these closer, the difference between the two is that in WISH the summands which are defined as the weights $w(x) = \prod_{\alpha \in \mathcal{I}} \psi_\alpha(\{x\}_\alpha)$, are each taken as the product of output of the factor ψ_α (which depends on the model) for all possible configurations $x \in \mathcal{X}$ (in our case $\mathcal{X} = \mathbb{Z}/2\mathbb{Z}^N$, so simply all binary strings of length N). On the other hand, the summands of the partition function in [4] are the product of the weights w_{ij} for all $i \neq j$ of all *subsets* S of the entire set of nodes, of a predetermined cardinality $|S| = m$. Comparing the weights defined in [4] with the factors (ψ_{ij}) of the weights of the Clique-structured Ising model in section 6 of [9], from [4] we have $w_{ij} = 1$ if and only if an edge between i and j exists, and [9] $\psi_{ij} = 1$ if and only if the random variables $\mathcal{X}_i$ and $\mathcal{X}_j$ have the same discrete value.

5.2 Results on Partition Function of Cliques

For a given graph $G = (V,E)$, the accuracy of the partition function of cliques algorithm described in [4] is decided by the choice of α below where we consider the modified adjacency matrix $W = (w_{ij})$ (which can be thought of as a weighted adjacency matrix) defined by:

$$w_{ij} = \begin{cases} 1 & \text{if } (i,j) \in E \\ \alpha & \text{if } (i,j) \notin E \end{cases}.$$

As mentioned above, the algorithm has computational guarantees only for these modified adjacency matrices.

The authors generated multiple sets Erdős-Rényi graphs for varying values of p and n, and the algorithm was run on these graphs. Due to the combinatorial difficulty of the problem, it was difficult to obtain exact data for the graphs in order to make comparisons. Thus, the authors relied heavily on results regarding the probabilistic connectivity thresholds of the Erdős-Rényi model described in the related work section. Some small instances of graphs (*i.e.* $G \sim \mathcal{G}(n,p)$ with small n) were sampled and $\text{Density}_m(G)$ computed. However these should be taken as toy examples and do not greatly reflect the practical applications of the algorithm in answering computationally difficult problems. In particular, for these cases the networkx package was able to determine the largest clique as well as the density functional readily, so there is no apparent gain to using a more optimal algorithm than the naive one given in the discussion in the related work.

5.3 Results on WISH Algorithm

5.4 WISH Algorithm

The WISH algorithm was implemented for the described Clique-structured Ising model. It is important to point out that the hardest part was the optimization with the constraint $A\sigma \equiv \mod 2$, where the developers of WISH assume they have access to an optimization oracle that can solve the constrained optimization problem.

In order to handle this constraint, we did a brute force search. This makes our implementation very inefficient, as for a fixed $|V| = n$ we go over $2^{n(n-1)}$ hash functions and a total of 2^{n^2} total combinations of A, b.

The authors of [9] did not consider the case of small graphs, for which WISH does not perform as one would hope for. The reason for this is that for small enough n, one needs to select the parameters of the algorithm appropriately in order to be definite that T solutions exist in order get the median weight, where T is defined in the algorithm. One simple counterexample is the following:

$$A = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix} \quad b = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$$

for which there does not exist a solution x to satisfy the constraint $Ax \equiv b \mod 2$.

5.5 Alteration of Clique-structured Ising model

One way to go around the issue of not having solutions satisfying the given constraint, is to redefine the distribution of the weights w_{ij} assigned to the pair of variables x_i and x_j. In [9] this weight is defined as being uniformly sampled from $[0, w\sqrt{|i-j|}]$ where w is a parameter set to 0.2. Since $Z \approx \sum_{\sigma \in \Sigma} w(\sigma)$, after some calculations we concluded that a sufficient condition would be to assign $w = \max(i,j) log_e 2$. This still only applies for large enough n, as we demonstrate in our results.

5.6 Study on Frequency of Non Existing Solution

A study was carried out on the frequency of how often it would be impossible to meet the constraint $Ax \equiv b \mod 2$. This was done through a simulation by increasing n and traversing all $i \in \{1, ..., n\}$, in order to go through all possible combinations of $A \in \mathbb{Z}/2\mathbb{Z}^{i\times n}$ and $b \in \mathbb{Z}/2\mathbb{Z}^{i}$. The following results took over 20 hours of computation on CAEN.

Percentage of non existing solution					
	n=1	n=2	n=3	n=4	n=5
i=1	0.7500	0.8750	0.9375	0.9688	0.9844
i=2	$*$	0.3906	0.4414	0.4697	0.4846
i=3	$*$	$*$	0.1985	0.2216	0.2351
i=4	$*$	$*$	$*$	0.1000	0.1110
i=5	$*$	$*$	$*$	$*$	0.0502

In Figure 2 are two plots drawn from the results of the fifth column above, where we assume the frequency of "faulty combinations" of pairs A, b which never satisfy the desired constraint can be approximated for higher values, based on $n = 5$. We observe through the rows of our table that such a a conclusion can be drawn (up to small difference for our calculations), with possibility that for each i as n increases, the frequency will be slightly higher.

The fitted equation on $n = 100$ is approximately $2e^{-0.72i}$.

5.7 Applications to the Planted Clique Problem

The plots depicted in Figure 1 represent two cases of the planted clique procedure described in the introduction: the first is a graph $G \sim \mathcal{G}(15, 0.1)$ with no planted clique. The second figure sees a planted clique of size 5 (K_5) placed on 5 randomly chosen vertices of the previously sampled G. The third figure represents the same graph G with a K_9 placed on 9 of the vertices. For comparison purposes, the authors placed the K_9 on 5 of the same vertices used in the previous figure. The algorithm was run on three similar cases and produced the results in Figure 3.

Here, the columns correspond to the graph G with either no planted clique, or a planted copy of K_4, K_6, or K_8. The algorithm was run with m chosen to match the planted clique. Note that the state of the art bound of $n^{1/2}$ obtained in [2] agrees with our data at least for small p. Namely, if we place a large clique of that size on the graph (here $n = 30$) there is a significant difference in terms of the density as compared to that computed when the graph has no clique.

Average Approximation using WISH over 10 simulations - Figure 4		
Length of String	Average approx. Regular model	Average approx. Altered model
N=2	13	13
N=3	26	24
N=4	42	41
N=5	67	63
N=6	89	91
N=7	116	115
N=8	146	145
N=9	191	198
N=10	232	248
N=11	330	329
N=12	380	387
N=13	596	575
N=14	686	685
N=15	1061	1177

5.8 WISH on Small Graphs

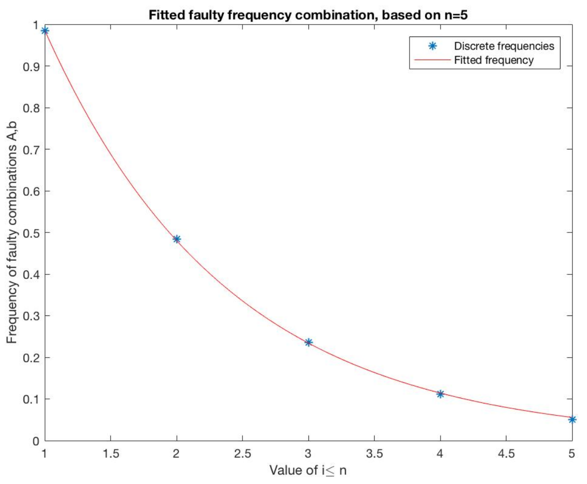
As was previously stated, computing the partition function is computationally expensive. For this reason, we were only able to run the WISH algorithm on small graphs, which unfortunately does not match well with our conclusion described in section 5.6. Depicted in Figure 4 are approximated values for the partition function based on the clique structured Ising model. Each data point corresponds to one of 10 trials for a fixed N representing the length of the string, where we take N from two to fifteen. We also provide a similar plot on the alteration of the clique structured Ising model we proposed, which does not yield much difference.

This is also clear from the corresponding average values for Figure 4 in the provided table, where for small sized graphs the values are very close to each other. We would need results from much larger graphs to draw meaningful results (rather than from a theoretical point of view) on whether or not what we propose would actually yield any improvements. Where the assigned weights are by default 0 (solution to constraint does not exist), we project those out.

We also looked into using an implementation provided for the MAP problem (Maximum a Posteriori assignment), though we did not use it at the end as it was difficult to adapt and generalize to the model we were dealing with.

5.9 Potential Equivalence

As future work, some investigation will be done into whether or not there is an alternative model



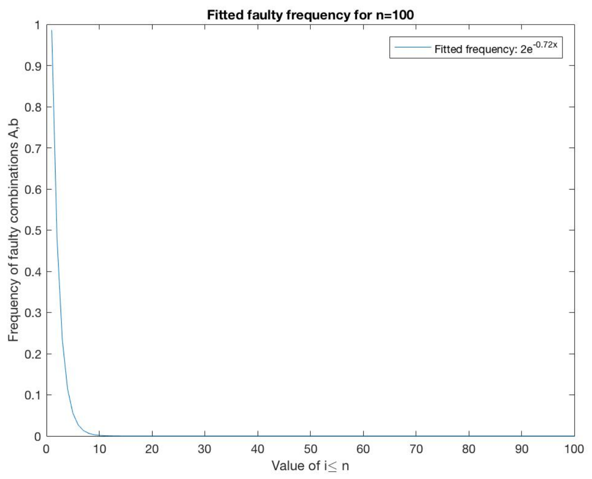


Fig. 2: To the left is a plot of how frequency of Hash functions of size $n = 5$ which cannot meet constraint, drops as i increases. To the right is a fitted plot for $n = 100$, expressed by $2e^{-0.72i}$.

Planted Clique				
p-values	G	$G+K_4$	$G+K_6$	$G+K_8$
p=0.01	0.7732	9.6226	13.5876	16.1301
p=0.02	0.9272	9.6235	13.5876	16.1301
p=0.10	1.7584	9.6283	13.5876	16.1301
p=0.50	7.1156	9.6612	13.5876	16.1301

Fig. 3: Table of Partition Function of Cliques Algorithm Run on Graphs with Planted Cliques

(other than the clique structured Ising model) which would allow an equivalence between partition functions to be shown. It is possible that such a change could render the WISH algorithm no longer valid so this investigation may not come to light. The author of [4] has suggested that considering bipartite graphs may offer a solution to the problem if a suitable Ising model could be found that lends itself to the WISH algorithm.

Alternatively, the problem could be phrased in a probabilistic sense: *i.e.* find a distribution of graphs which satisfy the clique structured Ising model, or another model. Then, with this distribution we can sample graphs (and their adjacency matrices). From this sample we can perform an explorative statistical analysis on this set of graphs using the Barvinok algorithm.

From the perspective of the WISH algorithm, we could change it such that we only consider subsets of a fixed size $m \lneq N$, so all the configurations in $\mathbb{Z}/2\mathbb{Z}^m$ which will also save a lot of computational time. We do not believe that in the case of the Ising model setting a constant weight as is done in [4] is ideal, as it defeats the purpose of the model, and the WISH algorithm might not work on such a model.

All in all, the main difficulty is that the algorithm in [4] does not take into account before hand the features of the graph itself, whereas in [9] the graph is being constructed from a given model. This poses the problem of coming up with a *factor graph* to adapt WISH to [4].

6 Conclusions and Future Work

The problem we dealt with has many potential improvements which could take place. Due to the time constraint, we were unable to complete everything we had hoped to. A major concern from the very start was complexity, which limited the conclusions we could draw through our simulations.

Our first major finding which conflicted a lot with what we proposed on doing at the beginning of the project, was that the two partition functions we describe are in fact not the same. While we initially knew the random setting of the WISH algorithm would lead to difficulty showing the equivalence, we were not expecting a total non equivalence (in the general setting) to be found. At the very start, we were prepared to explore both paths: showing that (1) was equivalent to (2) under randomness, and that in a deterministic setting, (2) is equivalent to (1). In the future we plan to look at specific cases of graphs where the two partition

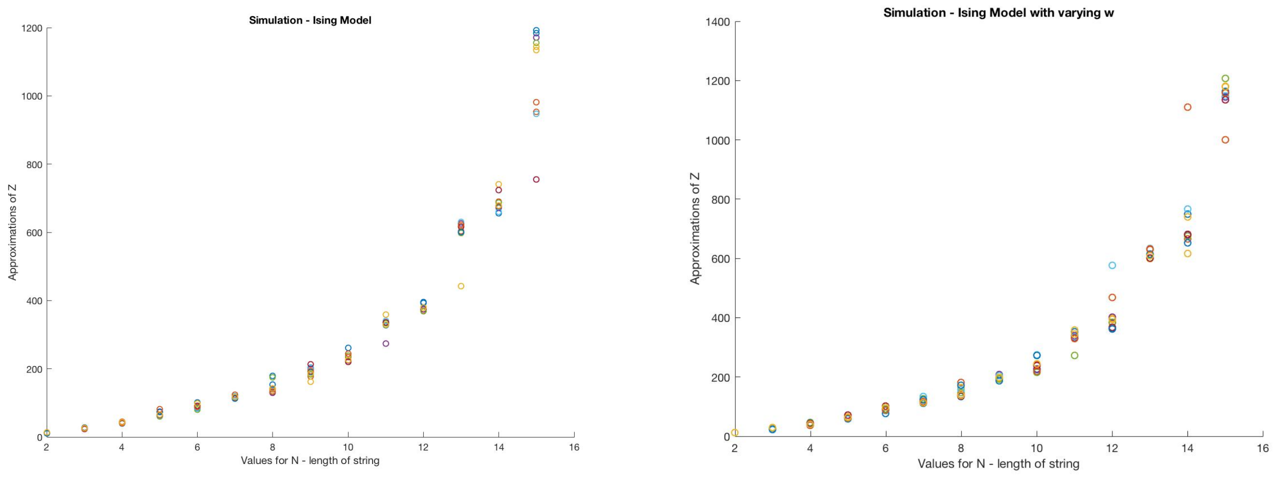


Fig. 4: To the left is a plot of a simulation of WISH on the clique structured Ising model, where we project out the assigned zero weights ${w_i}^t$. To the right is a similar plot, for a simulation on the alternation we proposed to the clique structured Ising model.

functions are equivalent. In particular, we would like to follow through on the suggestion to look at the case of bipartite graphs.

Regarding the partition function of cliques algorithm, we were unable to check our results against large graphs due to the computational difficulty of finding exact solutions to the density functional. The algorithm performed up to the standard as described in [4] on small graphs of size $n = 5$ however this shouldn't be considered evidence. Due to the mathematical proof behind the algorithm it stands to reason that the results should be valid. There is another potential application of the approximation to (1). The true incidence probability p could be estimated (in a statistical sense) by computing the approximation to the density $\text{Density}_m(G)$ of the graph for various values of m, and then comparing this with known bounds on the true value of $\text{Density}_m(G)$.

As for the WISH algorithm, it is clear from our table in section 5.6 that a much larger set of random variables need be considered, which is also evident from our plots in Figure 4. It would have been more useful if we tried using other optimization oracles, *i.e.* CPLEX in order to deal with this problem, though the computational problem in general would still be an issue.

References

[1] D. Achlioptas, "Stochastic integration via error-correcting codes."

[2] N. Alon, M. Krivelevich, and B. Sudakov, "Finding a large hidden clique in a random graph," *Random Structures and Algorithms*, vol. 13, no. 3-4, pp. 457–466, 1998.

[3] A. Barvinok, "Combinatorics and complexity of partition functions."

[4] ——, "Computing the partition function for cliques in a graph," *Theory of Computing*, no. Volume II(13), pp. pp. 339–355, 2015.

[5] ——, "Approximating permanents and hafnians of positive matrices," *arXiv preprint arXiv:1601.07518*, 2016.

[6] ——, "Computing the permanent of (some) complex matrices," *Foundations of Computational Mathematics*, vol. 16, no. 2, pp. 329–342, 2016.

[7] A. Barvinok and P. Soberón, "Computing the partition function for graph homomorphisms," *Combinatorica*, vol. 37, no. 4, pp. 633–650, 2017.

[8] S. Ermon, C. Gomes, A. Sabharwal, and B. Selman, "Optimization with parity constraints: From binary codes to discrete integration," *arXiv preprint arXiv:1309.6827*, 2013.

[9] ——, "Taming the curse of dimensionality: Discrete integration by hashing and optimization," in *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, 2013, pp. 334–342.

[10] V. Feldman, E. Grigorescu, L. Reyzin, S. Vempala, and Y. Xiao, "Statistical algorithms and a lower bound for detecting planted cliques," in *Proceedings of the forty-fifth annual ACM sympo-*

sium on Theory of computing. ACM, 2013, pp. 655–664.

[11] S. Hadjis and S. Ermon, "Importance sampling over sets: A new probabilistic inference scheme."

[12] O. J. Heilmann and E. H. Lieb, "Theory of monomer-dimer systems," in *Statistical Mechanics*. Springer, 1972, pp. 45–87.

[13] F. Krzakala, C. Moore, E. Mossel, J. Neeman, A. Sly, L. Zdeborová, and P. Zhang, "Spectral redemption in clustering sparse networks," *Proceedings of the National Academy of Sciences*, vol. 110, no. 52, pp. 20 935–20 940, 2013.