SPECTRA OF SPARSE GRAPHS AND MATRICES

ALEXEY SPIRIDONOV

ABSTRACT. We begin by briefly reviewing the essential results about sparse random graphs. We work primarily in the random graph model $G(n, \frac{\alpha}{n})$, with α constant. However, the main results of this paper generalize to sparse matrices with appropriate restrictions on the entries' distribution.

Let the ensemble of sparse matrices $\mathcal{S}_n$ be random graphs $G(n, \frac{\alpha}{n})$ endowed with independent random weights ξ, such that all moments of ξ are finite. Then, the moments of the spectrum of $\mathcal{S}_n$ are $\mathrm{E}\, M_{n,k} = \mathrm{E}\, \frac{1}{n} \operatorname{Tr} A^k$, where $A \in \mathcal{S}_n$. We show that $M_k = \lim_{n\to\infty} M_{n,k}$ exist and are finite for all k, and that these moments determine a unique spectral distribution λ. Moreover, for almost every sequence $A_n \in \mathcal{S}_n$, the spectral distribution of A_n converges weakly to λ as $n \to \infty$. Additionally, $M_{2k+1} = 0$, and the spectrum of $\mathcal{S}_n$ doesn't depend on the odd moments of ξ.

For matrices with bounded ξ, we show that M_{2k} lie asymptotically betwen $c_1^k A_k$ and $c_2^k A_k$, where A_k are the Bell numbers, and c_1, c_2 are constants depending on α and on the distribution of ξ. In the critical case $\xi = 1$, $\alpha = 1$, we get $c_1 = 1$, $c_2 = 4$; we also conjecture an improvement. The sequence M_{2k} for the critical case was registered as A094149 in Sloane's On-line Encyclopedia of Integer Sequences.

We present a number of numerical results, most significantly a reconstruction of the discrete part of the spectrum based on tree enumeration. We present a clear explanation of the elegant Li-Ruskey tree generation algorithm, to compensate for the dense presentation of the original paper. For our numerical calculations, we discuss and use a different tree generation algorithm with some original features, i.e. the ability to count the automorphism group sizes of the trees at negligible cost.

Based on the numerical work, we make a number of observations and conjectures. The most important one is that the lowest positive eigenvalue over all trees on $2k$ vertices is $2\cos(\frac{\pi k}{2k+1})$, over trees on $2k+1$ vertices it is $2\cos(\frac{\pi(2k-1)}{4k})$. We present some lemmas and hypotheses that may be of use in proving the statement. The significance of the lowest positive eigenvalue lies in quantifying the "gap" around 0 in the spectrum of random graphs with $\alpha \leq 1$. We conclude with a list of several interesting open questions.

1. Definitions and Outline

Definition 1.1. A *sparse matrix* (occasionally abbreviated *s.m.*) is an $n \times n$ random symmetric matrix with independent entries $a_{i,j}$, $1 \leq i \leq j \leq n$. $a_{i,j}$ is $\xi_{i,j}$ with probability $\frac{\alpha}{n}$ ($\alpha > 0$), and 0 otherwise. We assume that $\xi_{i,j}$ are identically distributed, independent random variables with finite moments, and that their distribution does not depend on n. The ensemble of such matrices be called $\mathcal{S}_n$.

We will distinguish two special types of sparse matrices:

Definition 1.2. A *sparse graph* (*s.g.*) is a sparse matrix with $\xi_{i,j} \equiv 1$. Although we place no restrictions on the diagonal elements (loops in the graph), we shall see that these make no contribution to the spectrum, and can be neglected. For n fixed, we call the ensemble $\mathcal{G}_n$; the dependence on the parameter α is always implicit.

Definition 1.3. A *bounded sparse matrix* (*b.s.m.*) has ξ with a bounded distribution (its moments grow at most exponentially). This ensemble is denoted $\mathcal{B}_n$.

We would like to characterize the limiting (as $n \to \infty$) spectral density of sparse matrices for various values of α, and for the three classes of matrices. $\mathcal{S}_n \supset \mathcal{B}_n \supset \mathcal{G}_n$, we will prove our results for as general an ensemble as we can.

This paper has four main sections. §2 discusses the history of the problem, summarizes previous work, and presents the results most important for the the rest of the paper. §3 answers basic questions about the moments of the limiting spectrum of all three types of sparse matrices. §4 describes the computational work done while exploring this problem, and suggests a number of questions. §5 discusses the part of the spectrum near 0.

2. Introduction

Sparse graphs and matrices have wide-ranging applications in mathematics and physics. They are used to model the energy levels of nuclei and to represent spin systems. They also come up in the modeling of networks; the spectra have been used to analyze the structure and reliability of these systems. Random matrix theory has connections with a plethora of other

subjects[1] and is somewhat older than the study of sparse systems specifically. The latter field originated in 1960, when Erdös and Rényi introduced the random graph model $G(n, p_n)$ – the graph on n vertices, with probability p_n of each given edge (total of $\binom{n}{2}$) being put into the graph; the choices are made independently (often the equivalent model $G(n, M_n)$ is used: all graphs with M edges are equiprobable). Our formulation of the problem in §1 generalizes the case $p_n = \frac{\alpha}{n}$ in the sense that we allow the edges to be weighted.

Erdös and Rényi (E-R) presented most of their classical results in one paper, [1]. In the following years, some of their results were extended, some estimates were improved or corrected, but the fundamental understanding of sparse random graphs was provided by this seminal paper. We follow their paper, sometimes giving extensions or modern formulations as in Bollobás [7]. E-R took n large, and studied what happened to the graph as p_n (or M_n) grew over time. In other words, the graph evolves by getting more and more new edges. It is natural to ask at every point in the evolution of G: how many connected components are there in the graph? what are the types of the connected components? how many components of each type are there? what is the probability of seeing a specific graph as a component or subgraph? E-R discovered that the evolution of the graph proceeds in essentially 4 distinct stages, in which some properties of the graph are substantially different.

The first stage is when $p_n = o\left(\frac{1}{n}\right)$; the present paper does not deal with this regime, as it is quite boring. Erdös and Rényi found that almost every graph (in the limit $n \to \infty$) consists of tree components. They determined threshold functions for the appearance of tree subgraphs of a given size size, and showed that the number of such trees (if p_n majorizes the threshold function) follows a Poisson distribution. It turns out that a.e. subgraph, which is a tree of the size in question, is in fact a component; the threshold for the appearance of trees of size k is $O\left(n^{\frac{k-1}{k-2}}\right)$.

In addition to trees, E-R calculated the threshold functions for some other simple types of subgraphs: complete, cycles, connected, etc. Bollobás [7] presents a considerably more general treatment of subgraph threshold functions, giving tools to find the threshold function of an arbitrary subgraph.

[1]One of the most exciting connections at this time is with the distribution of the zeros of the Riemann ζ function. See, for instance, M.L. Mehta, "Random Matrices" for a brief exposition.

We should note that non-tree components are essentially irrelevant to spectral problems: the fraction of vertices on these (we're disregarding the giant component) is $o(1)$ as $n \to \infty$. All of Bollobás's threshold function proofs are based on estimating the rth factorial moments of the T_k; the rth factorial moment $\mathrm{E}_r\, T_k$ is defined as $\mathrm{E}(T_k)_r = \sum_i (i)_r \mathrm{P}\ \{T_k = i\}$, where $(x)_r = x(x-1)(x-2)\dots(x-r+1) = \frac{x!}{r!}$. So, in the event that $T_k = i$, we are counting $(i)_r$, which is the number of distinct, ordered r-tuples of trees of size k, given that there are i trees. So, the r-th factorial moment produces the expectation of the number of such r-tuples. The factorial moments are important because the convergence of every rth factorial moment to λ^r implies weak convergence to the Poisson distribution with that parameter. (This is proved in Bollobás [7] using an elegant application of inclusion-exclusion) The factorial moments for a subgraph X with automorphism group size a converge to the expected number of ordered r-tuples of **vertex-disjoint** copies of X, which is

$$\mathrm{E}'_r(X) = \binom{n}{k}\binom{n-k}{k}\cdots\binom{n-(r-1)k}{k}\left(\frac{k!}{a}\right)^r p^{rl}\left(1 + O\left(rp\right)\right), \tag{1}$$

where the binomials signify the number of ways of picking the vertices of each graph in the r-tuple, the next factor adjusts for the number of ways one can select different edges to produce graphs isomorphic to X, the last factor gives the probability that all those edges are present. This quantity is $\sim \lambda^r$, with λ depending only on X. Showing convergence of $\mathrm{E}'_r(X)$ to $\mathrm{E}_r(X)$ is quite simple in the case of trees.

One result that holds in the regime $p_n = o\left(\frac{1}{n}\right)$ applies in two of the others as well, so we state it in full. While E-R proved the first version of this theorem, they did not estimate the rate of convergence, and used a less precise asymptotic for the variance. The following, stronger statement is due to Barbour (1982) [2]. Consider T_k, the number of components of $G(n, p_n)$ that are trees on k vertices. Then,

Theorem 2.1. *For fixed $k \geq 2$, any n and p_n, let*

$$\lambda_n = \mathrm{E}\, T_k = n^k \frac{k^{k-2}}{k!} p_n^{k-1}(1-p_n)^{nk}(1 + O\left(\frac{1}{n}, p_n\right)$$

$$\sigma_n^2 = \mathrm{Var}\, T_k = \lambda_n\left(1 + \lambda_n k^2(p_n + \frac{1}{n})\right) + O\left(1\right)$$

Then $\exists C(k)$, constant in n, such that,

$$\sup_{x\in\Re}\left|P\left\{\frac{T_k-\lambda_n}{\sigma_n}\le x\right\}\right|\le\frac{C(k)}{\sigma_n}$$

holds uniformly.

As for the case $k=1$, Barbour has an similar result that is valid when $p_n\sim\frac{c}{n}$. E-R's result (whether $k=1$ or not) is without estimates on the convergence rate, and holds $\forall k$ whenever $\lambda_n\to\infty$ as $n\to\infty$. For the purposes of our paper, the exact rate of convergence is immaterial. Barbour's proof involves a complicated analytic technique to achieve the rate of convergence of $\frac{C}{\sigma_n}$. E-R's proof is much simpler; they directly estimate the moments of the distribution, using the fact that T_k is the sum of an expression analogous to (1) with $r=1$ over all trees of size k. After some manipulations, using the properties of Stirling numbers of the second kind, the moments turn out to be normalized moments of a Poisson distribution. The moments, due to their normalization, converge to those of the Gaussian, which completes the proof.

The second regime E-R studied is that of $p_n=\frac{\alpha}{n}$ with $\alpha<1$. Then, the expected fraction of vertices on tree components (which can be calculated very easily using Theorem 2.1; we have a derivation in §4.3) remains 1, and a.e. large enough graph approximates it. However, there is now a non-trivial number of unicyclic components as well; they follow a Poisson distribution with a finite parameter independent of n. Almost every graph consists only of trees and unicyclic components, the largest trees (which are the largest components) are of size $O\left(\frac{1}{1-\alpha}\log n\right)$. The number of components, normalized by $\frac{1}{n}$, for a.e. large enough graph approximates $1-\frac{\alpha}{2}$. This is the so-called sub-critical regime; all trees are represented, and the structure of the spectrum is already quite interesting.

The third regime is the super-critical one, with $p_n=\frac{\alpha}{n}$, $\alpha\ge1$. As $\alpha\to1$, the size of the largest tree approaches $O\left(n^{\frac{2}{3}}\right)$. For $\alpha>1$, the largest component has size $g(\alpha)n$, where $g(\alpha)$ is a monotonic function growing from 0 at $\alpha=1$. (The largest tree component follows the same bound as with $\alpha\le n$: $O\left(\frac{1}{1-\alpha}\log n\right)$) This is the "giant component", which, as α grows, gradually absorbs all others. All but $o(n)$ of the remaining vertices still belong on trees, and the estimates on the number of tree components of various sizes continue to hold. Incidentally, it follows that $g(\alpha)$ is given by $1-\{\text{fraction of vertices on trees}\}$ (as estimated using Theorem 2.1, say).

With regards to the spectrum, this means that trees contribute a non-zero, but decreasing in α, fraction of the eigenvalues.

As $p_n \to \frac{\log n}{n}$, the graph becomes more and more connected; for $p_n \geq \frac{\log n}{n}$, almost every graph is connected, with the exception of $O(1)$ vertices. The trees disappear, from largest to smallest, when $\alpha = O(\log n)$: a graph with $p_n \sim \frac{\log n}{kn}$ almost surely has trees only of order $\leq k$. The distribution of the number of such trees is, again Poisson.

More recently, Rogers and Bray [5] showed that if $\alpha \to \infty$ (such as the logarithmic growth above), the spectrum of $G(n, \frac{\alpha}{n})$ approaches the Wigner semicircle with proper normalization in the limit $n \to \infty$. So, for all $\alpha \to \infty$, the limiting spectral distribution, normalized by $\frac{1}{\sqrt{\alpha}}$, is bounded. Another noteworthy recent result is due to Krivelevich and Sudakov [11], who showed that the largest eigenvalue in this model is $(1 + o(1)) \max(\sqrt{\Delta}, \alpha)$, with $o(1)$ decaying as $\max(\sqrt{\Delta}, \alpha) \to \infty$. Thus, while the normalized limiting spectrum is bounded, the normalized spectra for finite n are certainly not bounded as $n \to \infty$ (with $\alpha \to \infty$). Thereafter, Khorunzhy [6] proved that if $\frac{\alpha}{\log n} \to \infty$, and the model is modified to put zero-mean weights on the edges, the normalized distributions for finite n are a.e. uniformly bounded as well. In other words, putting zero-mean weights on the edges of a graph eliminates the gap between the bulk of the spectrum and the largest eigenvalue. We will see this in our numerical experiments in §4.1.

Remark. According to our Proposition 3.2, we can **always** modify ξ so that it has mean 0, and the same limiting spectrum.

Khorunzhy and Vengerovsky have a preprint [3] that explores the asymptotics of a random graph's limiting spectrum using moments, much like the present paper. However, we develop much finer estimates. Their paper features a fairly complex recursion relation for the moments of the limiting spectrum of a random graph with $\alpha = 1$. Finding a way to analyse that expression could lead to better asymptotics of the moments.

Physical methods (most of which are, unfortunately, not rigorous) have proven remarkably successful in analysing the spectrum of sparse matrices. The paper by Rogers and Bray [5] additionally shows that the asymptotic behavior for the density of the tail of the spectrum of $G(n, \frac{\alpha}{n}), \alpha$ finite, is $\left(\frac{x^2}{e\alpha}\right)^{-x^2}$; our asymptotic upper bound for the tails has essentially the right form, but has some unnecessary constant factors in both the base and the

exponent. Another remarkable result from a paper[2] Semerjian and Cugliandolo [13] derive an approximation for the spectrum of the giant component that is remarkably accurate numerically. Unfortunately, there is little hope that it will yield a mathematically rigorous result.

Up to this point we discussed primarily random graphs, for two reasons. The first is that, with suitable assumptions on the distribution of the matrix entries, many random graph results imply similar results about the sparse matrix spectrum. This is how we approach the problem in §3; "matrices" in the title of the paper should be considered an afterthought – all our results originate on graphs. The second reason is that random graphs have a beautiful theory of the discrete spectrum, which largely disappears (see the discussion of our numerical results in §4.1) when continuous random variables are chosen for ξ.

3. Moments of the Spectral Distribution

We wish to study the spectra of $n \times n$ sparse matrices as $n \to \infty$. In other words, we would like to study the random variable λ_n: a uniformly chosen eigenvalue (out of the n) for some matrix drawn out of $\mathcal{S}_n$. This is a well-defined construction for a finite n. Our first result is that the distribution of λ_n converges weakly to some λ. We approach the problem using the moments of λ_n. The kth moment is $\mathrm{E}\, M_{n,k} = \mathrm{E}\, \lambda_n^k = \mathrm{E}\, \frac{1}{n} \sum_{\lambda \text{ of } A} \lambda^k = \mathrm{E}\, \frac{1}{n} \operatorname{Tr} A^k$; thus, to get a handle on λ, we would like to study

$$\mathrm{E}\, M_k = \lim_{n\to\infty} \mathrm{E}\, \lambda^k = \lim_{n\to\infty} \mathrm{E}\, \frac{1}{n} \operatorname{Tr}(A^k).$$

Let $B = A^k = (b_{i,j})$; then $\mathrm{E}\, \frac{1}{n} \operatorname{Tr}(A^k) = \mathrm{E}\, \frac{1}{n} \sum_{i=1}^n b_{i,i}$. Thus, it is sufficient to calculate the diagonal elements of A^k:

$$b_{i_1,i_1} = \sum_{i_2,\ldots,i_k=1}^{n} a_{i_1,i_2} a_{i_2,i_3} a_{i_3,i_4} \cdots a_{i_{k-1},i_k} a_{i_k,i_1} \tag{2}$$

We will use this approach to address the following basic questions about M_k. Our proofs will be be primarily concerned with sparse graphs, but we discuss bounded and general sparse matrices as well.

(1) What are the requirements for the M_k to be defined and finite for all k? In other words, when does a limiting spectrum exist?
(2) Does the spectrum of a single large matrix approximate the limiting spectral distribution of the ensemble?

[2]Which also, amusingly, presents a physicist's re-derivation of some results by E-R.

(3) How fast does the density decay as $\lambda \to \pm\infty$?

3.1. **Finiteness of limiting moments.** Consider the subset of the terms of the summation for $\operatorname{Tr} A^k$ (see (2)), $S_l = \{a_{i_1,i_2}a_{i_2,i_3}\cdots a_{i_k,i_1}\}$, with the property that exactly l of the i_j are distinct. For instance, with $k = 3$, $n = 3$, $l = 2$, some valid terms are:

$$x_1 = a_{1,2}a_{2,2}a_{2,1},\ x_2 = a_{2,3}a_{3,2}a_{2,2},\ x_3 = a_{3,1}a_{1,1}a_{1,3}.$$

Clearly, the S_l partition the entire set of terms used in the summation.

Consider some $a_{i_1,i_2}a_{i_2,i_3}\cdots a_{i_k,i_1} \in S_l$. We label the l distinct indices $m_1, \ldots, m_l$ in the order they occur. So, for x_1, $m_1 = 1, m_2 = 2$, and we can write it as $x_1 = a_{m_1,m_2}a_{m_2,m_2}a_{m_2,m_1}$. Under this transformation, x_3 is written in the same way (with $m_1 = 2, m_2 = 3$): $a_{m_1,m_2}a_{m_2,m_2}a_{m_2,m_1}$, or 1 2 2 for short. Thus, we have an equivalence relation on S_l, which, in turn, partitions S_l into equivalence classes $C_{i,l}$ (i an index). We may now write

$$\mathrm{E}\, M_k = \sum_{l=1..k} \sum_{C_{i,l}\subset S_l} \lim_{n\to\infty} \frac{1}{n} \sum_{x\in C_{i,l}} \mathrm{E}\, x. \tag{3}$$

An equivalence class is the set of all terms gotten by plugging distinct m_i into an expression like this: $a_{m_1,m_2}a_{m_2,m_3}, a_{m_3,m_1}, a_{m_1,m_4}a_{m_4,m_1}$. However, we showed above a much more convenient notation; out of a_{m_i,m_j}, we only need to record i, since the rest of the information is duplicated. Then, our somewhat longer example becomes becomes 1 2 3 1 4 in the abbreviated form. Call an unordered pair of adjacent numbers in the abbreviated form a *step* (note that the last and the first entry should also be treated as adjacent). Let $D_{i,l}$ be the number of distinct steps in $C_{i,l}$; $D_{i,l}$ is 4 (out of a maximum of 5) in our example: 1 2, 2 3, 1 3, 1 4. Looking back at a term of the $\operatorname{Tr} A$ summation belonging to $C_{i,l}$, a step is just a reference to a particular entry of the matrix (pairs are unordered since the matrix is symmetric). Thus, every $x \in C_{i,l}$ uses l distinct indices and refers to $D_{i,l}$ distinct matrix entries. Finally, let d_j, $j = 1, \ldots, D_{i,l}$ be the multiplicities with which the distinct steps occur (in our example: 1, 1, 1, 2). Then, the contribution of x to the trace is given by:

$$\mathrm{E}\, x = \prod_{j=1}^{D_{i,l}} \frac{P_{d_j}\alpha}{n},$$

where P_k is the kth moment of the distribution of the ξ.

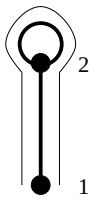


FIGURE 1. The walk corresponding to the equivalence class 1 2 2 1.

Since the sum of the trace goes over all combinations of indices, $|C_{i,l}| = n(n-1)(n-2)\cdots(n-l+1) = \frac{n!}{(n-l)!}$. For l fixed, this expression is $\sim n^l$ as $n \to \infty$. So, we may write

$$\mathrm{E}\, C_{i,l} = \lim_{n\to\infty} \frac{1}{n} n^l \prod_{j=1}^{D_{i,l}} \frac{P_{d_j}\alpha}{n} = \left(\prod_{j=1}^{D_{i,l}} P_{d_j}\alpha\right)\left(\lim_{n\to\infty} \frac{n^l}{n^{D_{i,l}+1}}\right) \tag{4}$$

Consider the abbreviated form of $C_{i,l}$: $v_1\, v_2\, v_3\, \ldots\, v_k$. We may view it as a closed walk of length k on a graph (with loops allowed) of l vertices (see our example in Figure 1).

Notice that $D_{i,l}$ specifies the number of edges used by the walk; a walk, by definition, uses edges that form a connected graph G. The expression in (4) has a non-zero limit only if $D_{i,l} + 1 \leq l$. This is only possible if G is a tree, and in that case, we have equality. Therefore, the limit expression in (4) converges to 0 or 1. Therefore, the $C_{i,l}$ that don't correspond to closed walks of length k on trees disappear from (3). We call the remaining equivalence classes T_l, where l designates the number of vertices in this class's trees. We can thus rewrite (3) as follows:

$$\mathrm{E}\, M_k = \sum_{l=1..k} \sum_{W_{i,l}\subset T_l} \prod_{j=1}^{D_{i,l}} P_{d_j}\alpha \tag{5}$$

This expression is independent of n, and the $P_{d_j}$are finite. Therefore, this expression is a finite constant for all k.

Proposition 3.1. *The M_k are well-defined and finite for all k. (holds for $\mathcal{S}_n$)*

The observation that the $W_{i,l}$ must go over trees gives a more surprising result:

Proposition 3.2. *$M_{2k+1} = 0$ for all k. (again, holds for $\mathcal{S}_n$)*

Proof. This follows from the fact that a closed walk on a tree must meet every edge an even number of times. Thus, it is impossible to have a closed walk with an odd length on a tree, and the sum in (5) goes over an empty set.

Note also that $d_j \equiv 0 \mod 2$ also means that only the even moments of ξ affect the limiting spectral distribution. □

Hence, the spectrum of a sparse matrix is always even.

We have shown that the moments of λ_n do converge. However, we do not yet know if these moments determine a distribution, or whether it is unique. This is exactly the classical moment problem; we have a sequence of moments $M_k \geq 0$, and we would like to know if there exists a unique probability measure with such moments. In 1922, Carleman proved that a sufficient condition is

$$\sum_{k=1}^{\infty} M_{2k}^{-\frac{1}{2k}} = \infty.$$

This requires an upper bound on the growth rate of M_{2k}, which we shall derive in §3.3.

3.2. **Limiting behavior of a matrix's spectrum.** λ_n is drawn randomly from the whole ensemble of eigenvalues of $\mathcal{S}_n$. This gives no guarantees about the behavior of individual matrices. Indeed, the ensemble could split into a number of large classes of matrices, with totally different spectral distributions that, when combined, give the moments that we observe. Below, we show that this cannot be, since $\lim_{n\to\infty} \operatorname{Var} M_k = 0$. Thus, a single large matrix approximates the limiting spectral probability; more formally, a.e. sequence of $\{\lambda_n\}$ converges weakly to λ.

We begin by computing $\operatorname{E} M_k^2 = \operatorname{E} \frac{1}{n^2} (\operatorname{Tr} A^k)^2$. Let $C = \{C_{i,l}\}$ be the set of all equivalence classes defined earlier. $C_{i,l} \in C$ can be viewed as a path on a graph, and we will single out $T \in C$, which are paths on trees. Multiplying the trace with itself term-by-term gives us the following:

$$\lim_{n\to\infty} \operatorname{E} M_k^2 = \lim_{n\to\infty} \frac{1}{n^2} \left(\sum_{\substack{T_1,T_2\in C,\\ \text{both trees}}} \sum_{\substack{s_1\in T_1,\\ s_2\in T_2}} s_1 s_2 + \sum_{\substack{NT_1,NT_2\in C,\\ \text{not both trees}}} \sum_{\substack{s_1\in NT_1,\\ s_2\in NT_2}} s_1 s_2 \right) \tag{6}$$

We argue similarly to §3.1. Consider the second term: say NT_1, NT_2 have l_1, l_2 distinct edges respectively. Without loss of generality, assume NT_1 is not a tree; then, it has $\leq l_1$ vertices; on the other hand, NT_2 has $\leq l_2 + 1$

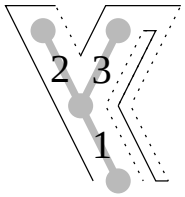


FIGURE 2. The walk $12b3bb13bb$. The "back" edges are dotted.

vertices. As in §3.1, the contribution of the s_1 and s_2 together is $\frac{c}{n^{l_1+l_2}}$ (c constant in n), but there are $\leq n^{l_1+(l_2+1)}$ choices of the pair (s_1, s_2). This means the contribution of the inside sum for every pair (NT_1, NT_2) is at most of the order of cn. As noted before, the number of equivalence classes of size k does not depend on n, so the second summand of (6) altogether is of the order of $\frac{C}{n}$, and thus vanishes in the limit.

The remaining term can be rewritten as:

$$\sum_{\substack{T_1,T_2\in C,\\ \text{both trees}}} \lim_{n\to\infty} \left(\frac{1}{n}\sum_{s_1\in T_1} s_1\right)\left(\frac{1}{n}\sum_{s_2\in T_2} s_2\right)$$

Each of the two products is of order of 1 in n, as we argued in the previous section. Hence, they can be split into a product of limits, which evaluates to (as in (4)) $(\mathrm{E}\,T_1)(\mathrm{E}\,T_2)$. Thus, we have:

$$\lim_{n\to\infty} \mathrm{E}\,M_k^2 = \sum_{\substack{T_1,T_2\in C,\\ \text{both trees}}} (\mathrm{E}\,T_1)(\mathrm{E}\,T_2) = \left(\sum_{T\in C} \mathrm{E}\,T\right)^2 = \left(\lim_{n\to\infty} \mathrm{E}\,M_k\right)^2$$

i.e. $\lim_{n\to\infty} \mathrm{Var}\,M_k = 0$.

3.3. **Asymptotics of the moments.** In order to estimate the speed of the distribution's decay, we would like to have some asymptotic estimates of the moments. We first bound the moments of matrices in the sparse graph ensemble $\mathcal{G}_n$. From the preceding discussion it follows that the $2k$th moment is exactly the number of closed walks of length $2k$ on plane rooted trees, in which new vertices/edges are encountered left-to-right. We can't have more than k edges, so these paths have the obvious interpretation as a proper subset of the $2k$-long strings of numbers from 1 to k (representing the edges).

A key observation is that the edges that move back towards the root are, for purposes of counting, indistinguishable. Consider the walk in Figure 2. The back edges, when written in string notation, are uniquely determined.

Therefore, we needn't label them at all; however, we do need the positions to distinguish between $12bb$ and $1b2b$.

First, we count the number of sequences without back edges (there is always a valid placement of the k back edges, as we shall see later). Thus, we want the number of k-sequences consisting of $1 \dots l$, $1 \leq l \leq k$, such that new numbers occur in order (equivalence classes have this property by construction); we call these *proper* k-sequences. This is equivalent to the number of ways of dividing k labeled objects into l unlabeled subsets. This number is exactly the Stirling number of the second kind, $S_{k,l}$. The desired total is therefore $\sum_{l=1}^{k} S_{k,l}$, which is shown in [4] to be asymptotically $A_k = \frac{b^k e^{b-k-\frac{1}{2}}}{\sqrt{\log k}}$, where b satisfies $b \log b = k - \frac{1}{2}$ (for comparison, this is asymptotically less than $\left(\frac{k}{\log k}\right)^k$).

Now, we are given k numbers, and need to insert back edges. Alternatively, we need to choose the placement of the k numbers into $2k$ slots – the empty slots will be back edges. That quantity would be $\binom{2k}{k}$; however, there is an extra constraint. There cannot be more back edges than labeled edges up to position i, for all i. In other words, the numbered edges and the back edges need to form a balanced parenthetical expression; the number of such expressions is exactly the kth Catalan number, $C_k = \frac{1}{(k+1)}\binom{2k}{k} \sim \frac{4^k}{(k+1)\sqrt{\pi k}}$.

This gives us an upper bound: $M_{2k} \leq \frac{4^k A_k}{(k+1)\sqrt{\pi k}}$.

To get a lower bound, notice that every proper k-sequence corresponds to a unique $2k$-walk on a star. To see this, insert a back edge after every real edge. For example, 1121232 becomes $1b1b2b1b2b3b2b$, which is a 14-walk on a 3-star. So, the number of equivalence classes is at least $\sum_{l=1}^{k} S_{k,l} \sim A_k$.

Thus, we have a fairly tight range for the asymptotic behavior of M_{2k}:

$$\operatorname{Lb} M_{2k} = A_k \lesssim M_{2k} \lesssim \frac{4^k A_k}{(k+1)\sqrt{\pi k}} = \operatorname{Ub} M_{2k}. \tag{7}$$

Numerical results from §4.2 suggest that the true growth rate is $A_k x^{\approx 2.3}$, so the upper bound could be substantially improved.

One hypothesis that would provide such an improvement is as follows. When a proper k-sequence uses all k vertices, all the C_k possible insertions of back edges are valid. It is easy to see that if a k-sequence uses $k-1$ distinct vertices, there are at most C_{k-1} valid ways of inserting the back edges. We believe that this can be generalized to sequences with l distinct vertices, to yield the following:

Conjecture 3.3. *M_{2k} is bounded from above by the Stirling transform of the Catalan numbers, $SC_k = \sum_{l=0}^{k} S_{k,l} C_l$, which is (by numerical estimates) $\approx (1.47^k + 1) A_k$.*

This is sequence A064856 in Sloane's encyclopedia [10], and an exponential generating function (albeit rather complex) is known. We believe it would be possible to come up with a good asymptotic upper bound for SC_k.

With or without the conjecture, $M_k = o\left(k^{\frac{k}{2}}\right)$, and so the Carleman condition (as discussed in §3.2) becomes:

$$\sum_{k=1}^{\infty} M_k^{-\frac{1}{2k}} \gg \sum_{k=1}^{\infty} \left(2k^k\right)^{-\frac{1}{2k}} = \sum_{k=1}^{\infty} (2k)^{-\frac{1}{2}} = \infty$$

Hence, M_k determine the distribution uniquely. Now, suppose that our matrices are drawn out of $\mathcal{B}_n$, with α not necessarily 1. Assuming that ξ is not identically zero, having a bounded distribution implies that $\exists c_1 > 0$ such that the moments $P_k \leq Ac^k$. Recall (5):

$$\mathrm{E}\, M_k = \sum_{l=1..k} \sum_{W_{i,l} \subset T_l} \prod_{j=1}^{D_{i,l}} P_{d_j} \alpha$$

Since $\sum d_j = k$, we have, $\prod_{j=1}^{D_{i,l}} P_{d_j} \leq \prod_{j=1}^{D_{i,l}} A c^{d_j} = A^{D_{i,l}} c^{\sum d_j} = A^{D_{i,l}} c^k$. Now consider $(A\alpha)^{D_{i,l}}$; if $A\alpha \geq 1$, this is bounded from above by $(A\alpha)$ and from below by $A\alpha$. The whole expression is then between $A\alpha(c^k)$ and $(Ac\alpha)^k$; if $A\alpha < 1$, these bounds are reversed. The important point is that in the case of general α, and matrices from $\mathcal{B}_n$, both $\mathrm{Ub}\, M_{2k}$ and $\mathrm{Lb}\, M_{2k}$ from (7) are modified only by CD^k, for some constants C and D.

This has several consequences; firstly, that means that the distribution is still determined by the moments. Secondly, the lower bound on the moments $M_k \gg k^{\frac{k}{2+\epsilon}}$ is super-exponential, so the distribution has infinite support. Finally, we can get non-trivial estimates of the asymptotic tail probabilities, both for $\mathcal{G}_n$ and $\mathcal{B}_n$. Let $P(u)$ be the limiting spectral measure; then,

$$\int_x^{\infty} dP(u) \leq \frac{1}{x^{2k}} \int_x^{\infty} u^{2k} dP(u) \leq \frac{1}{x^{2k}} \int_{-\infty}^{\infty} u^{2k} dP(u) = \frac{M_{2k}}{x^{2k}} \tag{8}$$

For simplicity of calculation, we bound A_k by $\left(\frac{k}{\log k}\right)^k$; we will specifically treat the case $A\alpha \geq 1$, so $M_{2k} \leq \left(\frac{4Ac\alpha k}{\log k}\right)^k$; will further write the constant simply as C. We wish to minimise (8), but we may minimize its logarithm

instead: $g_k = \log \frac{M_{2k}}{x^{2k}} = k\left(\log \frac{Ck}{\log k} - 2\log x\right)$. g_k has just one critical point in k, and a minimum at that, since both $X = x^{2k}$ and $Y = \left(\frac{Ck}{\log k}\right)^k$ are monotonic, Y grows much faster than X (and for fixed large x, there is an interval $(1, c)$ in which X is larger). The critical point is at (we discard $\frac{1}{\log k}$ since it's negligible for large enough k):

$$\begin{aligned}\frac{dg_k}{dk} &= \log\frac{Ck}{\log k} - \frac{1}{\log k} + 1 - 2\log x = 0\\ &\Rightarrow \log\frac{Ck}{\log k} = 2\log x - 1\\ &\Rightarrow \frac{Ck}{\log k} = \frac{x^2}{e} \text{ and } k \sim \frac{x^2}{Ce}\log\frac{x^2}{Ce}.\end{aligned}$$

In the last step, we used the fact that the Lambert W function $W(x)$, defined by $W(x)e^{W(x)} = x$, is asymptotically equal to $\log x$ (There are more precise asymptotics, but we choose to avoid the complexity).Therefore, our minimal upper bound for (8) is:

$$\frac{M_{2k}}{x^{2k}} = \frac{\left(\frac{Ck}{\log k}\right)^k}{x^{2k}} = \frac{\left(\frac{x^2}{e}\right)^k}{(x^2)^k} = e^{-k} \approx e^{-\frac{x^2}{Ce}\log\frac{x^2}{Ce}} = \left(\frac{x^2}{Ce}\right)^{-\frac{x^2}{Ce}}$$

It is possible to produce similar bounds for matrices in $\mathcal{S}_n$ that have ξ with sufficiently slowly growing moments, but this is an unenlightening technical exercise, and we refrain from doing so.

4. Computational Results

4.1. **Computing spectra.** The first numerical experiment we carried out was to compute and plot the spectra of large random matrices. To this end, we wrote a short C program that used LAPACK (`http://www.netlib.org/lapack/`) routines to compute eigenvalues (see Appendix A for the code). We computed the spectra of random graphs on 800 vertices for a number of sub- and super-critical ($<$ or $>$ 1) values of α. Actually, the results in Figure 3 were obtained by using graphs with equiprobable weights of ± 1 on the edges. However, as we pointed out in §3.1, $\xi = \pm 1$ and $|\xi| = 1$ produce equivalent limiting spectra.

The practical motivation for this weighing is that, for finite n and $\xi > 0$ (by Krivelevich-Sudakov) the highest eigenvalue comes either from the star of highest degree, or from the average degree; for relatively small values of α ($\alpha < \log^{\frac{1}{2}} n \approx 2.6$), the star dominates, and the square root of the degree

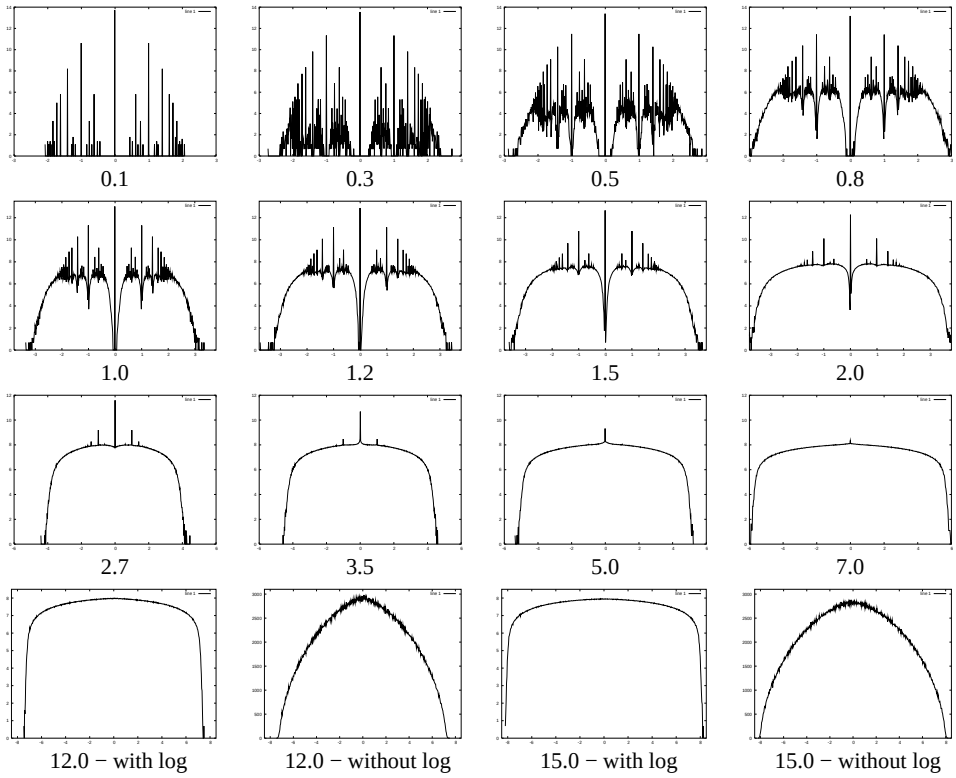


FIGURE 3. Numerical approximations of spectra of random graphs at different values of α; these are logarithmic-scale in y (to show detail) plots of histograms , with the exception of the $\alpha = 12.0$ and 15.0, which have both logarithmic and non-logarithmic versions. The horizontal scales are as follows: first row – $[-3, 3]$, second row – $[-3.8, 3.8]$, third row – $[-6, 6]$, fourth row – $[-8.5, 8.5]$.

lands well within the observed bulk: there are lots of other stars with similar degrees. However, when we transition to the super-logarithmic regime, the graph becomes almost regular, the average degree exceeds the square root of the maximum, and we see the associated a spectral gap: one eigenvalue of every matrix becomes detached from the bulk. This would lead to us seeing a bump of measure $\frac{1}{800}$ (very noticeable even on a linear scale) well to the right of the bulk. To avoid this extra noise, we chose to use a zero-mean random variable instead (in the spirit of [6]). (We did get experimental confirmation that $\xi = 1$ behaves exactly the same up to 2.6, and onwards, if one is to ignore the spectral gap) Later, we present two experiments with non-mean-zero r.v., which follow the prescribed pattern exactly.

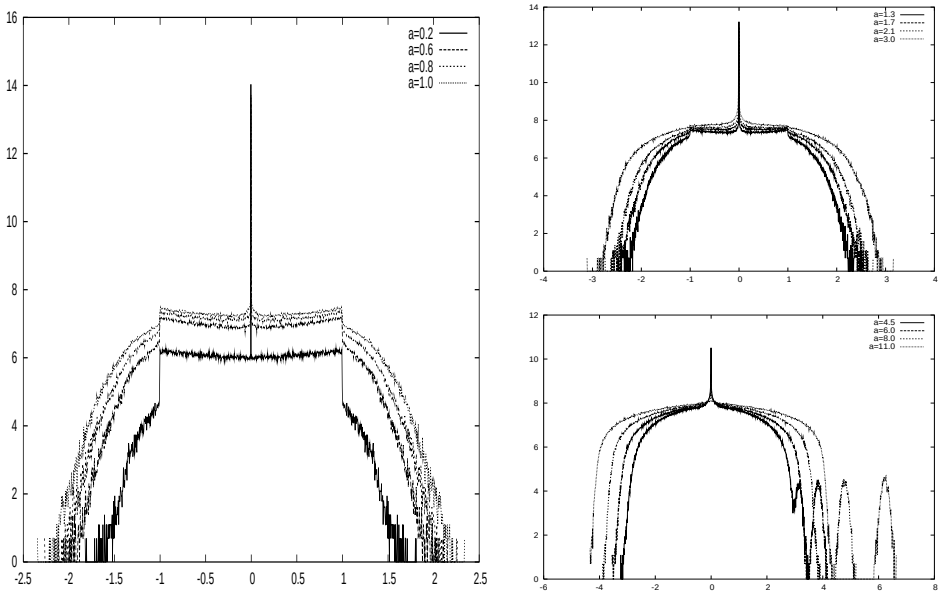


FIGURE 4. Numerical approximations of spectra of random graphs with uniform (in [0,1]) weights on edges at different values of α; all these are logarithmic-scale (to show detail) plots of histograms.

We clearly see discrete atoms comprising the entire spectrum for subcritical α, and then gradually see the giant component part of the spectrum gain in prominence. While the discrete part of the spectrum becomes indistinguishable at this resolution after $\alpha = 7.0$, it never goes away entirely, if $n \to \infty$ and α is constant (sublogarithmic, in fact, see [7]) relative to n. Observe that at $\alpha = 15$, the histogram resembles a semicircle as expected according to [5]. A somewhat unexpected feature of the graphs is the drastic dip near 0 for low values of α. A theoretical investigation of this phenomenon is presented in §5. Using our upper bound for the tails of the distribution, we consistently overestimate the expected ranges of the histograms by a factor of ≈ 5.6; we expect that a factor of 2 comes from our ovestimation of Ub M_{2k} by a factor of around 4^k; the other 2.8 must be due to the crude technique we used to compute the tail probabilities.

We performed similar experiments with non-trivial distributions for $n = 800$ (and $n = 600$ for some), and s ranging from 0.2 to 11. The three distributions are U_0^1 – the uniform density on $[0, 1]$, a sum of two independent U_0^1 (a triangular density from 0 to 2), and the standard normal distribution. The former is in Figure 4, while the latter two are in Figure 5a and b.

We discuss the coarse features of Figure 4, but a similar analysis is applicable to the others. For subcritical α, there is a prominent spike at zero, and a box-like protrusion between ± 1. The zero δ-function originates from primarily from disconnected vertices (which, as in r.g.s give a 0 eigenvalue); however, the 3-vertex path also has a zero eigenvalue, no matter the edge weights. The combined contribution, if computed according to §4.3, predicts the observed height of the spike well. Meanwhile, the box is produced by the single-edge components: a component with weight x gives eigenvalues $\pm x$, which perfectly matches the "box"'s location. All three small trees persist for supercritical α, but in reduced numbers. The giant component apparently also contributes some analog of these trees, as discussed in §4.5, as the peak and "box" in the supercritical case are larger than predicted by small trees alone.

As noted earlier, for super-logarithmic values of α, we observe an increasing spectral gap, with the largest-eigenvalue bump precisely $\frac{1}{n}$ in measure. The highest eigenvalue appears to approach $\alpha \operatorname{E} U_0^1$ as expected, but we didn't try α large enough to be sure. Another question that we cannot answer is whether/when the limiting spectrum becomes continuous in these cases; the answer probably requires a grasp of how dense the δ-functions in the spectrum of an r.g. are.

The case of the triangular density is very similar to the uniform one, and merits no further discussion. The Gaussian, predictably, gives smoother densities. Also as expected, the zero peak for various α is the same as that in the preceding cases.

4.2. **Computing moments.** We used an unsophisticated recursive search to calculate the first few M_{2k} for testing asymptotics, as well as to contribute entry A094149 to Sloane's encyclopedia [10] of integer sequences. The first 13 M_{2k} are: 1 3 12 57 303 1747 10727 69331 467963 3280353 23785699 177877932 1368977132. Our program searches for ordered-first-occurrences number sequences that have $l-1$ distinct steps, and use l numbers (formulation from §3). It uses a few simple constraints to make searches reasonably fast. Refer to Appendix A for the code. The main constraint for computing further terms with the program is that simply counting them will take too long. Khorunzhy's [3] recurrence relation should be able to go farther if implemented properly.

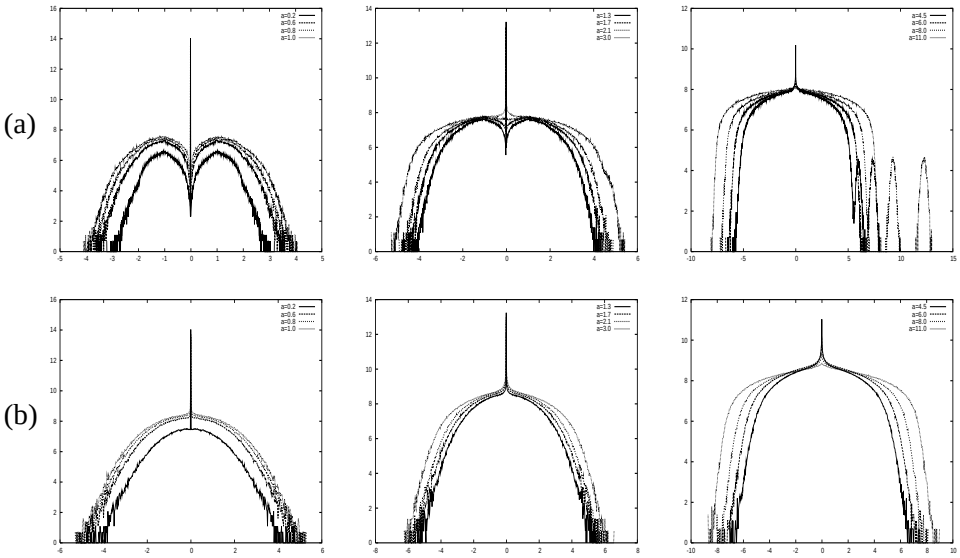


FIGURE 5. Numerical approximations of spectra of weighted random graphs on edges at different values of α; all these are logarithmic-scale (to show detail) plots of histograms. The top row's weights are sums of two [0,1] uniform variables. The bottom row's weights are standard Gaussian.

4.3. **The discrete spectrum.** From the previous sections, we know that a sparse graph consists of a giant component (for $\alpha \geq 1$ only), and a large number of small tree components. The contributions of non-giant, and non-tree components to the spectrum vanish in the limit $n \to \infty$. Since there are countably many trees, and each gives only a finite number of eigenvalues, there is a countable set of eigenvalues produced by small trees (size bounded a.e by $O(\log n)$, except at $\alpha = 1$). We call this the discrete spectrum. By Theorem 2.1, if T_k is the number of tree components of size k, then the distribution of $\frac{T_k - \mathrm{E}\, T_k}{\sqrt{\mathrm{Var}\, T_l}}$ rapidly converges to $N(0,1)$, the standard Gaussian, where

$$\mathrm{E}\, T_k \sim n \frac{k^{k-2} \alpha^{k-1} e^{-k\alpha}}{k!}$$

$$\mathrm{Var}\, T_k \sim \mathrm{E}\, T_k \frac{1 + (\alpha - 1)(k\alpha)^{k-1} e^{-k\alpha}}{k!}$$

Note that the variance is linear in n. Thus, if we consider experiments producing $n \times n$ matrices, $\frac{T_k}{n}$ will rapidly converge to a constant as $n \to \infty$. Another observation is that $\frac{\mathrm{E}\, T_k}{n}$ decays quite rapidly in k for all α. Therefore,

	α					
K	0.4	0.8	1.0	1.3	1.7	3.0
∞	1	1	1	0.5770	0.30882	0.05952
10	0.9976	0.8898	0.7545	0.52866	0.30469	0.05952
20	≈ 1	0.95182	0.82403	0.55961	0.30848	0.05952
25	≈ 1	0.96519	0.84217	0.56553	0.3087	0.05952
30	≈ 1	0.97403	0.85566	0.56916	0.30878	0.05952
50	≈ 1	0.99036	0.88779	0.57493	0.30881	0.05952

TABLE 1. $\mathcal{F}_K$for different values of K and α; $K = \infty$ denotes the fraction of eigenvalues accounted for by tree components.

if we compute the eigenvalues of T_k for k small, we will have a good lower bound on the discrete spectrum. Every tree in T_k has k eigenvalues, so each class contributes $n\frac{k^{k-1}\alpha^{k-1}e^{-k\alpha}}{k!}$ eigenvalues. There are n eigenvalues total, and the fraction of the eigenvalues given by trees up to size K is therefore

$$\mathcal{F}_K = \sum_{k=1}^{K} \frac{k^{k-1}\alpha^{k-1}e^{-k\alpha}}{k!}.$$

Table 1 shows values of $\mathcal{F}_K$ for several values of K and α. As we see, we can capture a very reasonable fraction of the eigenvalues given by trees by working at $K = 25$ (the highest value achievable with only a few hours of computation time).

We will iterate over all $t \in U_k$, the set of unlabeled trees of size k. For a given t, we compute: a – the size of its automorphism group, and $\lambda_1, \ldots, \lambda_k$ – its eigenvalues. Then, t occurs $\frac{k!}{a}$ times in the set of k-vertex labeled trees, so the expected number of copies of t in our r.g. is $w_t = \frac{k!}{ak^{k-2}} \mathrm{E}\, T_k$ (where k^{k-2} is the number of labeled trees). Thus, we add $\lambda_1, \ldots, \lambda_k$ to the distribution as δ-functions of weight w_t. This distribution, when computed on U_k, $k = 1 \ldots K$, will be our lower bound for the spectrum.

Our implementation uses a new tree enumeration algorithm, which provides automorphism group sizes. The eigenvalues are simply combined into a high-resolution histogram. The latter decision (over more fine-grained storage – like hashing and combining the delta-functions) makes for easy implementation, and easy comparison with numerical results from the §4.1.

4.4. **Tree Generation.** We will list unlabeled, unrooted (free) trees – what graph theorists usually mean by a tree. The number a_n of such trees of size n was first studied by Cayley and later by Pólya, who produced a generating function for a_n, and a general apparatus for enumerating sub-classes of trees.

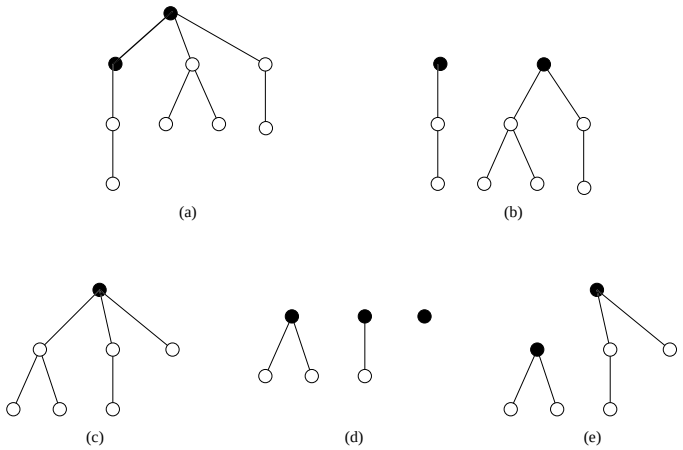


FIGURE 6. Illustrating how canonical rooted form trees break into arbitrary rooted trees. Black vertices denote the roots.

Otter proved in [8] that $a_n \sim \frac{BA^n}{n^{5/2}}$, with $A \approx 2.9557653$ and $B \approx 0.5349496$. The sequence has no known nice form, and so it is perhaps surprising that one can enumerate the corresponding objects in $O(a_n)$ time – the best possible. We will present two algorithms for generating trees. Refer to Li-Ruskey [9] for a history of the tree generation problem.

The observation underlying both algorithms is that a free tree can be made rooted in a canonical way. Consider the following definition:

Definition 4.1. The *center* C of a graph G is the set of vertices v such that the maximum distance (along a path) from v to any other vertex achieves the smallest value possible in this graph:

$$C(G) = \{v | v \in V(G), \max_{w \in V(G)} d(v,w) = \min_{a \in V(G)} \max_{b \in V(G)} d(a,b)\}.$$

It is an easy fact that all trees have $|C(T)| = 1$ or 2, such trees are called *unicentral* and *bicentral*, respectively. If we have a unicentral tree T, selecting the center as the root gives a unique rooted tree (the *canonical rooted form*). In the bicentral case, the trick is to use some ordering of the rooted trees (we shall obtain one later). Then, let T_1 and T_2 be the rooted trees gotten by making either of the two center vertices the root. We define the canonical rooted form of T to be the larger of T_1, T_2 in our ordering. With that, we have a 1-1 correspondence between free trees and canonical rooted trees.

Hence, it is enough to generate canonical rooted trees. Suppose T is a canonical rooted form of a bicentral tree (refer to Figure 6a for an example).

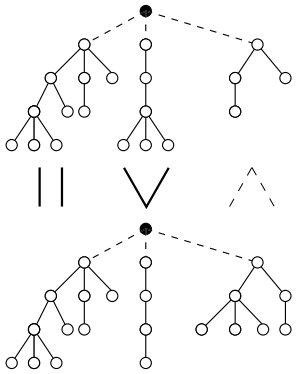


FIGURE 7. Comparing two trees of height 4. The two largest subtrees are equal, and so the second largest subtrees dictate that the top tree is larger; the last (dashed) comparison isn't actually done.

Delete the edge separating the two centers; let each center be the root of the corresponding component (Figure 6b). Note that we must get two rooted trees of the same height[3]. Conversely, we can glue together any two rooted trees of the same height to get a bicentral tree, with the original roots as the two centers. Using our ordering, we pick the correct center as the root for the composite tree, and we have a canonical rooted form (CRF). Similarly, to produce[4] unicentral trees, we delete the edge between the root and one of the tallest components, getting two trees of heights differing by one (Figure 6e).

Thus, our problem is reduced to generating unlabeled rooted trees with height restrictions, and gluing them back together. This is what the two algorithms do differently.

4.4.1. *Listing trees: the Li-Ruskey algorithm.* The algorithm we present first is due to Li and Ruskey [9], and is optimal both in terms of speed – $O(a_n)$ and space – $O(n)$ (which is negligible, considering that it isn't practical to do computations on, i.e., the $\approx 3.5 \times 10^{14}$ trees of size 40). It is not the first algorithm with these properties, but it is much shorter than the predecessors. Nonetheless, it is rather tricky to understand, and the original paper does a poor job of explaining it. This paper attempts to rectify this.

[3]Height is exactly what it sounds like – the largest distance from the root to another vertex.

[4]This way of decomposing bicentral and unicentral CRF is used by Li-Ruskey. Our algorithm just deletes the root, as in Figure 6d.

The first step is to introduce an ordering on rooted trees. There are two equivalent formulations of the rule. As an algorithm (assuming $T_1 \neq T_2$): if $height(T_1) < height(T_1)$, then $T_1 < T_2$; otherwise, compare the subtrees (hanging from the root) going from largest to smallest in this ordering (applied recursively). Then we say that the smaller of the two trees is whichever of T_1, T_2 has a smaller subtree first. An illustration of the comparison process is in Figure 7.

For the other statement, we need to define a parent array. First, we define the *parent* of vertex i: the adjacent vertex on the path from i to the root. Analogously, the *children* of the vertex i are the neighbors that are not on the path from i to the root. Consider a tree T of size n, labeled with the numbers $1 \dots n$. Then, the parent array is the sequence of numbers $a_1, \dots, a_n$, where a_i is equal to the label on i's parent; the parent of the root is labeled "0". Figure 8b shows a parent array for the labeling in Figure 8a. We also define a depth-first labeling of a tree: we traverse the tree, starting at the root, labeling previously unvisited vertices $1 \dots n$ sequentially, in the following manner. If the current node i has any unvisited children, we pick one, and visit it (recursively). Otherwise, we return to the parent of the current node. (for example, see two labelings in Figure 8a, c)

To get our ordering, consider all of T's depth-first labelings. Pick the one that has the maximum (lexicographally) parent array; this will be the parent-array representation, $par(T)$. Then, $T_1 < T_2 \Leftrightarrow par(T_1) < par(T_2)$, where parent arrays are compared lexicographically. The process of finding, and comparing parent-array representations (PARs) is illustrated in Figure 8. Observe that the outcome of the comparison is the same as that of the previously described depth-based recursive comparison algorithm.

The PAR is associated with a specific *depth-first* labeling (which, noted before, is obtained by a recursive traversal of a tree: start at the root, and upon arriving at any vertex, visit all its children before returning to the parent). We can use this labeling to draw trees in a canonical way. The rule is to draw the tree so that the depth-first traversal visits the children of every vertex from left to right (making it a pre-order traversal). Equivalently, the subtrees of every vertex are drawn in decreasing order fom left to right. Figures 6 and 7 show trees in this format, as will most figures to follow. This way of drawing makes it easy to tell which of two trees is larger.

The first key idea for the algorithm is that we can construct a tree $\mathcal{T}$ of all rooted trees. Observe that if we remove the last entry of $par(T)$ (denoted

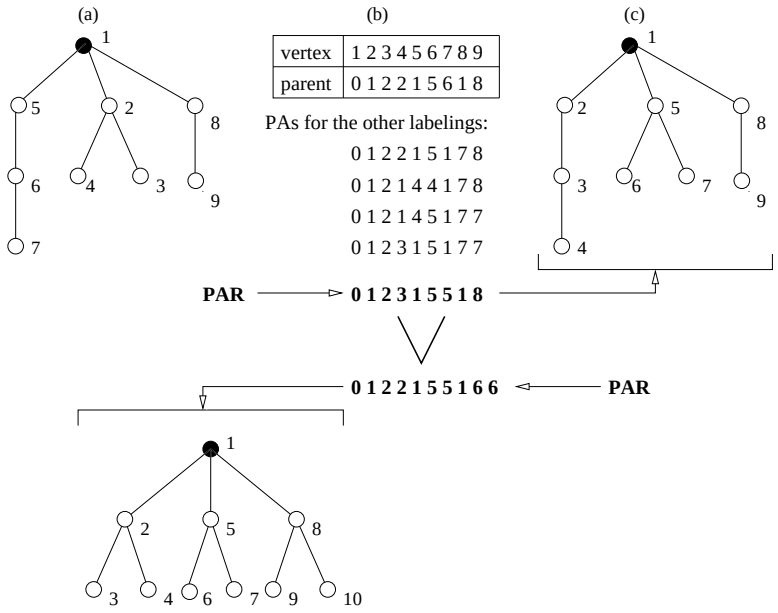


FIGURE 8. The top half of the figure lists the parent arrays resulting from all non-equivalent depth-first traversals of the given tree. The largest of these is marked by boldface type, and PAR, denoting that it is the parent-array representation of the tree. The lower half of the figure shows another tree and its PAR, which, despite having more vertices, is less than that of the first tree.

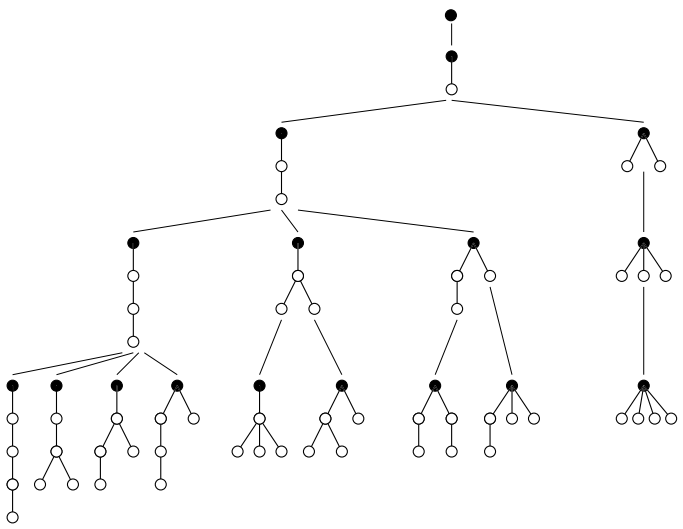


FIGURE 9. The tree of rooted trees up to size 5. Children of each rooted tree are arranged in decreasing order from left to right.

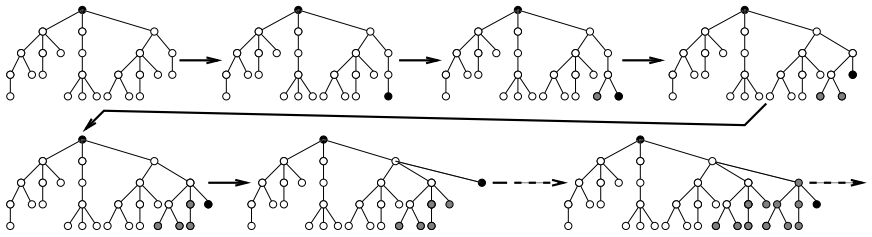


FIGURE 10. The result of repeatedly adding new vertices at the lowest possible point, ρ_T. Black circles denote the root and the last added vertex. Gray circles denote all new vertices.

$par(1 \dots n-1)$, the parent array for vertices $1 \dots n-1$), we still have a PAR of a smaller tree T'. Therefore, we can consider T' to be the parent of T. Clearly, by a sequence of such steps, every PAR is reduced to just 0, the parent array of the single-vertex tree. We thus get the tree of all rooted trees in Figure 9; notice that we arranged the children of every vertex from largest to smallest, going left to right.

We now want to traverse $\mathcal{T}$ in a simple way. To do this, we consider the relationship between the children of a given tree on n vertices (i.e. the children of the 3-path in Figure 9): we see that, from left to right, the vertex n is attached progressively higher along some path to the root. Indeed, suppose we start with a tree T on $n-1$ vertices, and can get a PAR on n vertices by appending p to $par(1 \dots n-1)$. Then, $par(1 \dots n-1)+par(p)$ is also a PAR – when n moves up, all the (right-most at every level) subtrees that contained it become smaller. In other words, the action of picking up the last (labeled n) vertex, and moving it up one towards the root creates another valid PAR with the prefix $par(1 \dots n-1)$. Therefore, to get all the children of a tree, we need only find the lowest (farthest from the root) valid point ρ_T for attaching a vertex.

So, to traverse $\mathcal{T}$, we need to know:

(1) How ρ_T changes when we append a vertex to T at that point.
(2) How ρ_T changes when we shift the last vertex up, towards the root.

To answer the first question, consider what happens when we keep adding a vertex to a tree as low as possible. This is illustrated in Figure 10. We observe that this procedure repeatedly copies a certain subtree. (Excluding the sequence of paths on the very left of $\mathcal{T}$, which is a degenerate case) While we shall not formally prove it, it is quite straightforward to check that the subtree T_c is determined by the following requirements:

(1) Let t be the root of T_c. Then t's parent p_c is on the path from ρ_T to the root. t is the next-to-rightmost child of p_c. (so, the root of smallest subtree that isn't the one under construction)
(2) The subtree T' rooted at the rightmost child of p_c (the copy of T_c under construction) **can** be completed to become T_c. More formally, $par(T')$ is a prefix of $par(T_c)$.
(3) T_c is such that p_c is as close to the root as possible, while fulfilling 1 and 2.

Say that T_c tree is rooted at t, and has size s. If we have just put in the final vertex in the current copy of T_c, then $\rho_T = p_c = par(t)$, since we are starting a new copy. The current copy is complete only when vertices $n-s \dots n-1$ are all in it, with $n-s$ being the root; when the current copy is incomplete, $n-s$ is a non-root vertex in the previous copy. (The arithmetic follows from our labeling scheme). When $n-s$ is the root of a T_c copy, $par(n-s) = p_c$, which is less than s, being its parent. (in all other cases, $par(n-s) \geq s$) So, if $par(n-s) < s$, we have $\rho_T = par(t)$.

If we haven't just finished a copy of T_c, we just want to copy the next vertexs. The counterpart (in the previous subtree) of the vertex we just added (labeled $n-1$) is $n-s-1$; in the previous subtree, $n-s$ was added next. Therefore, ρ should be the counterpart of $par(n-s)$ in the current subtree. So, $\rho = par(n-s) + s$.

Remark. If p_c is the parent of several subtrees isomorphic to T_c, it does not matter which one we use in the above algorithm.

It remains to see what happens to ρ_T when the last vertex (labeled n) gets shifted up towards the root (question 2). We will figure out what happens to t and s after this shift, and use the preceding discussion to get ρ. Let $p = par(n)$ before the shift. Then, after the shift, $par(n) = par(p)$, and the tree rooted at p clearly satisfies criteria 1 and 2 for being T_c. Suppose there is a vertex p', higher up the path than $par(p)$, with a subtree T'_c also satisfying these criteria. Then, its rightmost subtree T' (the presumed tree under construction) is its prefix. However, by shifting n up, we have made T' smaller than what it used to be. Since it's currently a prefix, T' prior to the shift was larger than T'_c, which is a contradiction. Therefore, $t = p$, and $s = n - t$.

Putting all of this together, we get the elegant piece of code in Algorithm 1. It clearly spends a constant amount of time on each node of the tree $\mathcal{T}$ (the loop goes over the children). Therefore, it runs in time $O(a_n \leq n)$,

Algorithm 1 A C implementation of the Li-Ruskey algorithm for listing rooted trees of order N. It is invoked as `gen(1, 0, 0)`; no initialization is necessary.

```
void gen(int n, int t, int s) {
  if (n > N) print_it();
  else {
    if (s == 0) par[n] = n-1; /*left-most path in T*/
    else {
      if (par[n-s] < t) par[n] = par[t]; /*new copy*/
      else              par[n] = c + par[n-c];
    }
    gen(n+1, t, c); /*add a vertex at rho*/
    while (par[n] > 1) { /*move the last vertex up*/
      t = par[n];
      p[n] = par[t];
      gen(n+1, t, n-t);
    }
  }
}
```

which, using Otter's asymptotics is $O(a_n = n)$. It only uses $O(n)$ storage, and the call stack has the same memory requirement. Thus, this code is optimal in both time and space. Its only limitation is that the enumeration is restricted to a specific order (decreasing in our ordering). Random-access algorithms (akin to Prüfer codes) for unlabeled tree enumeration are the major open problem in the subject. Unfortunately, it appears very difficult, since the number of unlabeled (rooted or free) trees of a given size does not have a natural correspondence with any nice set.

We now return to free trees; our treatment may differ somewhat from Li-Ruskey. It is also not as detailed as the preceding discussion, since this extension of the algorithm is a matter of implementation, rather than conceptual development. Refer to the Li-Ruskey paper [9], or their production program available at: `http://www.theory.csc.uvic.ca/~cos/inf/tree/FreeTrees.html` for working implementations. Recall that, given a canonical rooted form of a tree T, we can delete the edge from the root to the largest (left-most) subtree (as in Figure 6b and e). We then get two rooted trees T_l, T_r such that $height(T_l) + 1 = height(T_r) = height(T)$ if T was unicentral, and $height(T_l) = height(T_r) = height(T) - 1$ if T was bicentral.

Suppose we are asked to produce all free trees on n vertices. The strategy will be to use Algorithm 1 to generate T_l with appropriate restrictions, and

then, instead of calling `print_it()`, to generate T_r in a similar manner. The maximum height of an n-vertex CRF is $\lfloor \frac{n}{2} \rfloor$, so T_l will have height at most $\lfloor \frac{n}{2} \rfloor - 1$. Note that if $height(T_l) = h$, then T_r will have height at least h as well, and will require $\geq h + 1$ vertices. Thus, Algorithm 1 requires some modifications, so that it neither exceeds the available number of vertices, nor the available height in the search. Observe that the height only grows in the very left-most path down $\mathcal{T}$, when `s == 0`. We add two parameters: `m` for available size, and `h` for available height, update them appropriately, and, whenever either reaches 0, we proceed to generate T_r.

Suppose T_l was of height h and used m vertices. Then, T_r must have height $h + 1$ (unicentral) or h (bicentral), and use the remaining $n - m$ vertices. We modify Algorithm 1 very similarly, except that that we first walk down the left-most path in $\mathcal{T}$ to the depth we need, and then only perform "add vertex" operations that do not increase depth.

In the unicentral case, we must have the largest subtree of T_r be at most (in our ordering) T_l. To solve this, we arrange for T_l to be produced as the left-most (first in the parent array) subtree of T, then set the modified Algorithm 1 to complete the tree (with the root of T as the root of T_r). This works, since the algorithm only produces canonical trees.

On the other hand, in the bicentral case we want to produce a CRF, which means that (the entire) $T_r \leq T_l$. This is also easy enough to achieve: to the unfinished T containing T_l, we add a vertex at the root, to act as the root of T_r. Then, the algorithm again produces the desired tree, and a simple manipulation on the parent array is needed to remove the fake root.

The realization of these ideas for gluing rooted trees is tricky and tedious; it is probably easiest to start with two separate routines (one producing T_l, and calling the next to produce T_r). However, both will have Algorithm 1 as the skeleton, and can be combined with some care. Both implementations cited above use a single recursive routine.

We will not argue why the added constraints for producing free trees do not affect the running time of the algorithm. The main observation is that there are no late-pruned dead ends in the recursion: if some branch of search tree doesn't contain any valid trees, we know that at its root.

4.4.2. *Listing trees: constrained concatenation.* Prior to finding the Li-Ruskey algorithm, we had designed a simpler one. It also runs in linear time, but it stores a substantial number of trees. Without going into the detailed analysis (which, as we shall see, would involve counting trees of a number of sizes and

n	15,21,27	16,22,28	17,23,29	18,24,30	19,25,31	20,26,32
used	625	1321	2822	6115	1.3×10^4	3×10^4
×savings	5.05	5.86	6.85	7.95	9	11
used	7×10^4	1.6×10^5	4.0×10^5	9.5×10^5	2.3×10^5	1.2×10^7
×savings	12	13	14	15	17	20
used	2.7×10^7	6.3×10^7	1.4×10^8	1.4×10^8	3.3×10^8	8.1×10^8
×savings	23	27	31	37	45	49

TABLE 2. The approximate number of trees our algorithm stores to compute all trees up to size n, compared with the number of trees of size $n-1$.

depths), Table 2 gives rough estimates of the space requirements for various tree sizes. Given that our program uses 16 bytes per tree, trees up to order 30 (of which there are $\sim 2.3 \times 10^{10}$) are easily accessible on modern workstations. On the other hand, performing non-trivial computations on that many trees would take on the order of months. Thus, for our purposes, this method is perfectly practical. The table shows space savings as compared to an algorithm by Read referred to in [9]. That algorithm is viable up only to $n = 25$, so we've achieved a substantial improvement.

The main merits of our approach are as follows: it is practical, it is constructive (as compared to Li-Ruskey), it gives the size of the automorphism group of our trees (something we need) practically for free, it is fast (takes 2 minutes to count trees of size 27), and it is relatively easy to implement.

Consider, as before, the canonical rooted form of a free tree on n vertices. Now, delete the root vertex. We are left with a number of rooted trees. Let the two largest trees be T_1 and T_2, and suppose T_1 has height h. Since we are dealing with a CRF, either $height(T_2) = h$, or $height(T_2) = h-1$. Suppose T_1 has height h. Then, T_1 and T_2 together use at least $2h+1$ vertices; hence $2h+1 \leq n-1 \Rightarrow h \leq \frac{n}{2} - 1$. Additionally, no component can have more than $n-1-h$ vertices (minus the root, minus the size of T_2). So, we want all rooted trees of height 1 having $1 \ldots n-2$ vertices, trees of height 2 having $1 \ldots n-3$ vertices, ..., trees of height $\lfloor \frac{n}{2} \rfloor - 1$ having $\lceil \frac{n}{2} \rceil$ vertices. This is the quantity estimated in Table 2.

To make these rooted trees, we will use an inductive construction. First, we define an ordering. Suppose an order on rooted trees of size $< n$ has been defined, and $size(T_1) = n$. We use the following ordering: $height(T_1) < height(T_2) \Rightarrow T_1 < T_2$; otherwise $size(T_1) < size(T_2) \Rightarrow T_1 < T_2$; otherwise

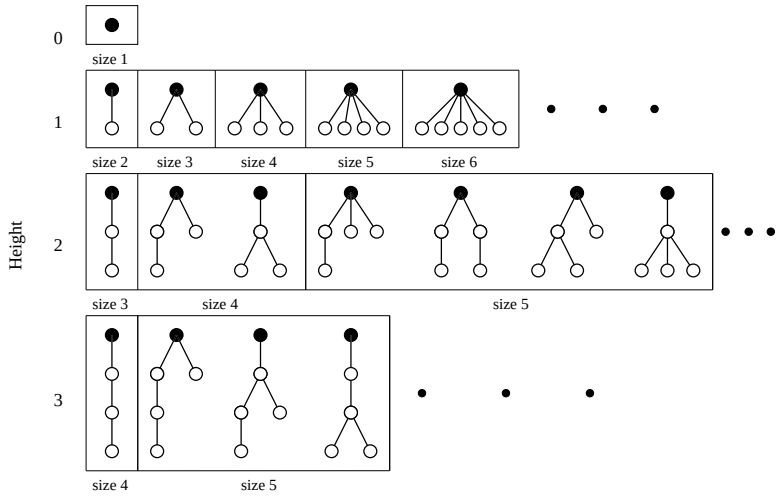


FIGURE 11. The ordering used by the constrained concatenation algorithm (if read top-to-bottom, left-to-right).

delete the roots of both, and compare the components from largest to smallest (lexicographically). Figure 11, read from top to bottom, left to right, illustrates the rule.

Now assume that we have all required trees up to size $n-1$, organized thus: for each height, we have a list of trees in increasing order. (exactly as in Figure 11). As an optimization, we keep for every height an index of where each size begins. This allows easy access to trees of a specific height and size, and, if one traverses the list sequentially, the index tells us what size tree we are at (so we don't have to store it).

To build a tree of size n, we select a largest component (trying each in order, from the one-vertex tree to the maximum available depth and size). This reduces the available size, and possibly height. We select the next component, so that it fits in the available size and height, and does not exceed the previous component. We try the valid trees in increasing order also. We repeat this until we run out of vertices (and we can have $n-1$ vertices on components). Thus, we get a non-increasing sequence of subtrees, which clearly is a canonic representation for a rooted tree. Notice that in this algorithm, we never try components that don't work. Hence, it is linear in the number of trees we enumerate. A compact implementation is shown in Algorithm 2. It is invoked with increasing values of n, with trivial updates to the size index between invocations. The algorithm uses a bit string to

Algorithm 2 The constrained concatenation algorithm for rooted trees. The `Tree` type is simply a bit string (say 64-bit integer). `l[h]` is the list of trees of height `h`, in order. `si` is the size index: `si[h][n]` gives the index in `l[h]` at which the size `n+1` begins; it is updated between calls to the routine.

```
void rootedsearch(int n, int cur_h, int max_h,
                  int max_i, Tree t)
{
  if (n == 0) { listAdd(l[cur_h], t << 1); }
  else {
    int i, h, maxi, maxh = min(max_h, n - 1);
    for (h = 0; h <= maxh; ++h) {
      int size = h + 1, new_h = max(cur_h, h + 1);
      if (h < max_h) maxi = si[h][n];
      else           maxi = min(si[h][n], max_i);
      for (i = 0; i < maxi; ++i) {
        if (i == si[h][size]) ++size;
        rootedsearch(n - size, new_h, h, i + 1,
                     t << (2*size) | l[h]->data[i]);
      }
    }
  }
}
```

store a parenthetical expression representing the rooted tree[5]. See Appendix A for a complete program using this routine.

Using this approach, the size $\mathcal{A}$ of the root-stabilizing automorphism group of a rooted tree is very easy to get. As we are composing the list of subtrees $T_1, \ldots, T_k$, we know the size of each automorphism group $a_1, \ldots, a_k$ (by induction). All of these give independent (in the direct product sense) automorphisms of T, which contributes $\prod a_i$ to $\mathcal{A}$. We can have additional automorphisms if $T_i = T_{i+1}$ for some i. Since T_i are added in increasing order, it is very easy to find how many duplicates of each there are. Suppose there multiplicities are $m_1, \ldots, m_l$, $\sum m_i = k$; then, $\mathcal{A} = \prod a_i \prod m_i$. To compute this requires a very simple modification to Algorithm 2, which is incorporated into the final program in Appendix A.

Moving on to free trees, we use exactly the same approach as for rooted trees: look at the tree without the root, and produce an ordered list of

[5]For example, the first size 2, height 5 tree in Figure 11 has parenthetical representation ((())()()), corresponding to the bit string 1110010100. So, the bit arithmetic in the code simply concatenates such strings to record the addition of a new component. This compact realization of trees was chosen for ease of storage and speed of concatenation (arrays would be much more unwieldy, despite the flexibility).

subtrees. However, we are generating a canonical rooted form, so there are extra constraints. Firstly, we require a special case to make sure that the largest two subtrees don't differ in height by more than one. Secondly, a bicentral tree must use the larger of its two representations. This means that if we delete the edge connecting the two centers, getting T_l (the largest component) and T_r (the rest), we must have $T_l \geq T_r$. To achieve this in the code, after choosing T_l, we create the list of its subtrees $T_1, \ldots, T_k$ after deleting the root. Then, while adding more components (which will comprise T_r) to T, we check that the sequence never exceeds T_i lexicographically. This means keeping a flag for whether this constraint is still necessary – the requirement disappears as soon as we add the first subtree to T_r that's strictly smaller than the corresponding T_i. If the constraint is still necessary, we walk our data structure (`l`, `si`) up to the available size, depth, not exceeding the previous component, and not exceeding the current T_i. These two extra requirements make the algorithm produce valid CRF trees. Computing $\mathcal{A}$ is exactly the same as for rooted trees, except that if bicentral trees are symmetric around the center, they need an extra factor of 2. Symmetry means exactly that T_i = the corresponding subtree of $T_r$for all i; so, we multiply by 2 if this constraint checking was necessary till the end of the construction of the tree.

The extra constraints added for free tree generation don't add any dead branches in the search tree, and take a constant time to check for every free tree produced. Therefore, this algorithm runs in linear time. See Appendix A for the a complete, working implementation.

4.5. **Spectrum lower bounds.** Using our constrained concatenation algorithm, we iterate over all t, compute their eigenvalues using LAPACK, compute the contribution of these eigenvalues as in §4.3, and add these to a histogram. We present a comparison of the lower bound with the full spectrum in Figure 12 for a supercritical and subcritical value of α. For $\alpha = 0.8$, the difference between the spectra (c) is very noisy (all the noise of the Monte-Carlo approximation is made visible), since it is plotted on a scale $\approx 150\times$ smaller than (a) and (b). It also shows that a larger fraction of the missing density comes from the fringes (the plot is less curved); we need larger trees to get these eigenvalues. For $\alpha = 1.7$, the scale is reduced only $\approx 7\times$, and so the noise isn't much amplified. Rather, we have cancelled out the vast majority of the discrete spectrum (99.9%), so what remains comes from the giant component.

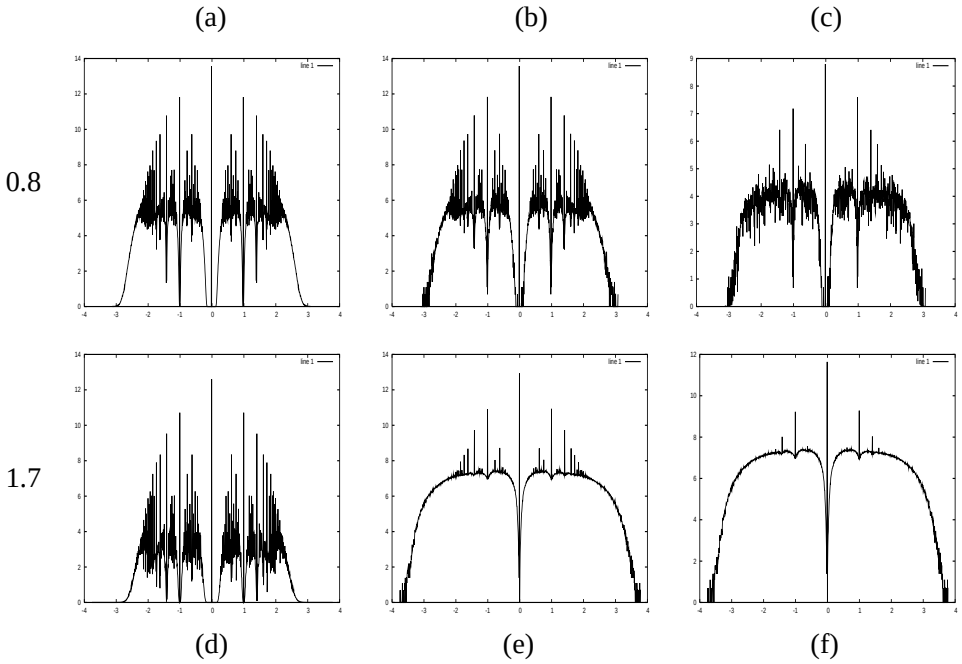


FIGURE 12. Comparing the full spectrum with the lower bound of the discrete spectrum for two values of α: 0.8 and 1.7. All plots are on a natural-log scale in y, with $x \in [-4, 4]$. In (a), (b), (d), (e), $\log y \in [0, 14]$; in (c), $\log y \in [0, 9]$; in (f), $\log y \in [0, 12]$. (a) & (d) are the lower bound plots for $K = 25$ and 23 respectively. (b) and (e) are full spectra; (c) and (f) are differences between the two preceding ones.

Despite lacking tree components, the histograms in (f) has large spikes at $0, \pm 1, \pm\sqrt{2}, \ldots$, much like the discrete spectrum. Informally, $\alpha = 1.7$ is not long after the emergence of the giant component; at this stage, we expect it to locally look like a tree, having branches that are exactly small trees. Then, eigenfunctions that restrict their action to that branch would produce the spikes we observe. A similar effect might be responsible for the drop in density around 0. We conjecture that for all α constant in n, the limiting spectrum of the giant component has a non-zero discrete part, resembling the discrete spectrum produced by trees. To the best of our knowledge, this problem is completely unexplored. It would also be interesting to find in what growth regime of α the discrete spectrum of the g.c. disappears. Here, we conjecture that it will happen at, or before the disappearance of the tree discrete spectrum at $\alpha \sim O(\log n)$. Proceeding in the same vein, is the

elimination of the two discrete spectra sufficient for the limiting distribution to become continuous with respect to the Lebesgue measure?

4.6. **Other observations.** On perusing the numerical results, we noted several other regularities which warrant theoretical treatment. The spectrum of trees is always symmetric; proving this is trivial. Observe that trees are bipartite (pick a $v \in T$, there is a unique path from v to any other $u \in T$; divide the vertices into two sets based on the value of $length(v, u) \mod 2$). Given an eigenvalue λ of a bipartite G with eigenfunction e_v, reverse the sign of e_v on one of the two parts; that gives a new eigenfunction with eigenvalue $-\lambda$.

If one considers the smallest positive eigenvalue among all trees of size k, it turns out to occur on paths for k even, and on paths with a 1-edge split at the end ($- \cdots - \begin{smallmatrix} / \\ \backslash \end{smallmatrix}$) for k odd. This enables to calculate what this eigenvalue is, precisely. We treat this observation in §5.2. A final remark is that the spectrum of any tree of size k appears to necessarily included in the spectrum of another tree of size $k+4$. Proving this conjecture might be interesting.

5. The Neighborhood of 0

Figures from the previous section show a prominent atom at 0, and a spectacular drop in the height of delta-functions as one approaches zero from either side. The zero spike was analysed by Bauer and Golinelli. As for the decline in height, we can develop a theoretical (conjectural at the moment) upper bound on the δ-functions of the discrete spectrim near 0 by following the numerical observations.

5.1. **Number of zero eigenvalues on all trees of a fixed size.** In [12], Bauer and Golinelli derive an explicit formula, a generating function and an asymptotic approximation for z_k – the number of zero eigenvalues (with multiplicity) in all trees of size k. Their approach uses two methods. First, they characterize the number of zero eigenvalues $Z(F)$ in a given forest F using a specialization of the method in §5.2.1. This is quite easy; however, they then use this characterization to cleverly rewrite $Z(F)$ as a sum of a simple quantity over induced subtrees of the forest. Armed with that identity, the rest of the paper uses fairly standard techniques from enumerative combinatorics to produce its results.

According to this paper, the expected multiplicity of the zero eigenvalue in a tree of size k is $\mathrm{E}\, Z_k = (2x_* - 1)k + \frac{x_*^2(x_*+2)}{(x_*+1)^3} + O\left(\frac{1}{k}\right)$. As before, the expected number of trees componets of size k in $G(n, \frac{\alpha}{n})$ is $n\frac{k^{k-2}\alpha^{k-1}e^{-k\alpha}}{k!}$. So, the asymptotic (in k) contribution of trees of size k to the δ-function at zero is $(2x_* - 1)\frac{k^{k-1}\alpha^{k-1}e^{-k\alpha}}{k!} + \frac{x_*^2(x_*+2)}{(x_*+1)^3} + O\left(\frac{1}{k}\right)$. We could have used this (or even the explicit formula) to improve our estimates in §4.5, if the extra precision was worth the effort.

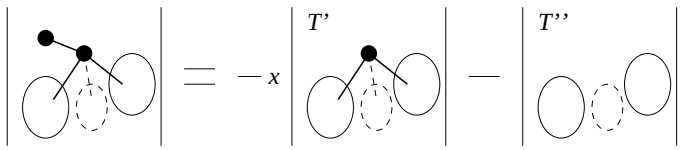
5.2. **Relation between tree size, and least positive eigenvalue (LPE).** The basic problem is: over the collection of trees on k vertices, what is the non-zero eigenvalue of least magnitude? (by the symmetry of the tree spectrum identical to the least positive eigenvalue) Once we have this information, we can use the estimate on T_k from Theorem 2.1 to predict the maximum measure of an atom in $(-\epsilon, \epsilon)$ for small $\epsilon > 0$.

5.2.1. *Characteristic polynomials of trees.* In this section we introduce a (fairly standard) tool for working with characteristic polynomials (CPs), and prove some easy lemmas (with the goal of proving that the LPE occurs on walks/forked walks). We will call the coefficient of x^i of a polynomial a_i.

Find a degree 1 vertex (which always exists) in the tree, and expand the determinant by the row corresponding to it (without loss of generality, rearrange the matrix to the presented form):

$$\begin{vmatrix} -x & 1 & 0 & \dots & 0 \\ 1 & -x & * & & \vdots \\ 0 & & \ddots & & \\ \vdots & & & \ddots & \vdots \\ 0 & \dots & & \dots & -x \end{vmatrix} = -\lambda \begin{vmatrix} -x & * & \dots & \\ \vdots & \ddots & & \\ & & \ddots & \vdots \\ & & \dots & -x \end{vmatrix} - \begin{vmatrix} 1 & * & \dots & \\ 0 & -x & \dots & \\ \vdots & \vdots & \ddots & \vdots \\ 0 & & \dots & -x \end{vmatrix}.$$

The first determinant in the right-hand side is clearly the CP of the tree with the degree 1 vertex deleted (called T'). The second rhs determinant can be expanded along the first column to get, again, a CP of a subtree (now with two vertices deleted, called T''). The relation is much more comprehensible in pictorial form:



The CP of the (however many) disconnected components is quite obviously the product of the components' CPs. The above relation turns out to be surprisingly useful in the proof of the following results.

Proposition 5.1. *In a CP of a tree of size k, $a_i = 0$ if $a_i \neq k \mod 2$. Moreover, the non-zero terms have alternating signs:* $\operatorname{sgn} a_k = (-1)^k$, $\operatorname{sgn} a_{k-2} = (-1)^{k-1}$, $\operatorname{sgn} a_{k-4} = (-1)^{k-2}, \ldots$, $\operatorname{sgn} a_{k-2j} = (-1)^{k-j}, \ldots$.

Proof. By induction. The single-vertex tree I has characteristic polynomial $-x$, satisfying both conditions.

Suppose that the claim holds for all trees up to size $< n$. Then, consider a tree of size n, and expand its CP using the above relation. Consider the second term of the rhs, $\operatorname{cp} T''$: it is the product of CPs of smaller trees, with a total of $n-2$ vertices. We need to show that CPs of two trees satisfying the claim multiply properly. Suppose A and B are CPs from trees of size $l, m < n$, with coefficients a_i and b_i; then the non-zero terms in AB will have power of parity $l + m \mod 2$. Additionally,

$$\operatorname{sgn} ab_{l+m} = \operatorname{sgn} a_l \operatorname{sgn} b_m = (-1)^{l+m};$$

$$\operatorname{sgn} ab_{l+m-2} = \operatorname{sgn} a_{l-2} \operatorname{sgn} b_m \oplus \operatorname{sgn} a_{l-2} \operatorname{sgn} b_m = (-1)^{l+m-2};$$

$$\operatorname{sgn} ab_{l+m-2j} = \bigoplus_{i=0}^{j} \operatorname{sgn} a_{l-2i} \operatorname{sgn} b_{m+2i-2j} = (-1)^{l+m-2}.$$

$x \oplus y$ simply signifies that if x and y have the same sign, then $x + y$ has the same sign (otherwise, it would be undefined, but we don't have this problem). So, indeed, two CPs fitting the claim multiply to produce another polynomial fitting the claim. Hence, $\operatorname{cp} T''$ has non-zero terms only with powers of parity $(n-2) \equiv n \mod 2$. The signs on the terms are: $\operatorname{sgn} a_{n-2-2j} = (-1)^{n-2-j}$; however, the expression is negated, so the signs become: $\operatorname{sgn} a_{m-2(j+1)} = (-1)^{n-(j+1)}$, which is what we want. □

The first term of the rhs is just $(\operatorname{cp} I)(\operatorname{cp} T')$, of size 1 and $n-1$. By induction, and by the preceding discussion, the product is of the right form for a tree of size n. Clearly, the sum of the two parts fits the claim as well.

Corollary 5.2. *CPs of odd trees have a factor of x.*

Lemma 5.3. *If a tree has even size and $a_0 \neq 0$, then $|a_0| = 1$.*

Proof. Again, we proceed by induction. The CP of the two-vertex tree is $x^2 - 1$.

Suppose the CP of all even-sized trees $< n$ either has a factor of x, or has $|a_0| = 1$. Given a tree of size n, we expand its CP as before. The first term has a factor of x, and so makes no difference. If any of the forest's components has a factor of x, $a_0 = 0$. Otherwise, they all have $|a_0| = 1$, and so does the forest, and so does the whole tree.

It's very easy to come up with examples to show that the analog for odd trees and a_1 does not hold. □

5.2.2. *Conjectures about the LPE.* This section explores possible ways of proving that among trees of size $2k$, the LPE comes from the path, and for $2k + 1$, the LPE comes from the forked path. In this section, speculation abounds, and claims shouldn't be taken as fact unless said otherwise.

Suppose that we have an integer-coefficient polynomial of degree $2n$, with $a_0 \neq 0$, and having only even powers. Moreover, suppose that its signs are fixed so that a_0 is positive, $a_2$is negative, etc. Notice that both even and odd tree CPs can be brought into this form by appropriately multiplying by -1 and dividing by x. If we wish to make it have a root as close to zero as possible, while having limitations on the magnitude of the coefficients, we can use the following argument. a_0 should be 1, since otherwise we would need a greater derivative (and greater coefficients) to get to zero faster. a_2 should be as negative as possible. We want $|a_4|$ to be as small as possible, $|a_6|$ as large as possible, etc. However, a_2 is most important, since the remaining terms have a factor of x^4or more, which reduces their contribution drastically for small x.

Conjecture 5.4. *The minimum possible magnitude of any $a_i \neq 0$ for a CP of a tree is exactly $i + 1$.*

This was observed from computation. This means that odd trees can't actually get $a_0 = 1$ when reduced to the above form. This complicates matters substantially; moreover, it means that trees of size $2k + 1$ needn't necessarily give a lower LPE than trees of size $2k$. We used a program in Appendix A to show LPE($2k + 1$)>LPE($2k$) for trees of size ≤ 25; this is probably true in general (assumingConjecture 5.6bonus, this follows from the results of §5.2.3). So, odd trees are altogether unnecessary for the bounding the decay of peaks near zero. From now on, we only talk about even trees.

Conjecture 5.5. *For a CP of a tree of size $2k$, $\left|a_{2(k-i)}\right| \leq \binom{2k-i}{2(k-i)}$. This maximum is achieved $\forall i$ by the $2k$-walk. (The second part is easy to prove, see §5.2.3).*

This might be useful, but to show that the path produces the LPE, it is essential to have control over the ratio $\frac{a_{i+2}}{a_i}$; especially for the first i such that $a_i \neq 0$. The terms of higher order of smallness (with extra factors of x^4 and larger) should be easier to control. Clearly, the control over the ratios afforded to us by Conjectures 5.4 and 5.5 is insufficient. Estimating upper bounds on the ratios between adjacent non-zero coefficients is a probably a good direction to explore in order to prove:

Conjecture 5.6. *The LPE on trees of size $2k$ occurs on the $2k$-path.*

***Bonus**: The LPE on trees of size $2k+1$ occurs on the forked $2k$-path.*

5.2.3. *Conditional calculation of the LPE.* Here, we assume that Conjecture 5.6 is true, and calculate the eigenvalue explicitly. To do this, let $b_1, \ldots, b_{2n}$ be an eigenfunction of the eigenvalue λ for the $2n$-path. Then, $\lambda b_{k-1} = b_k + b_{k-2}$ with the constraint that $b_2 = \lambda b_1$ and $b_{2n-1} = \lambda b_{2n}$.

Suppose $\lambda = 0$; then, $b_2 = 0$, so $0 \cdot b_3 = b_2 + b_4 \Rightarrow b_4 = 0, \ldots, b_{2n} = 0$. But, then $b_{2n-1} = 0$ as well, and, propagating backwards: $0 \cdot b_{2n-2} = b_{2n-3} + b_{2n-1} \Rightarrow b_{2n-3} = 0, \ldots, b_1 = 0$. So, 0 is not an eigenvalue. We also want to show that $b_1 \neq 0$. Suppose otherwise; then $b_2 = 0$, so $\lambda \cdot 0 = 0 + b_3 \Rightarrow b_3 = 0$, etc. Thus, we can renormalize all our eigenfunctions to have $b_1 = 1$, and $b_2 = \lambda$.

Finally, suppose $\lambda = 2$; then $b_2 = 2$, $2 \cdot 2 = 1 + b_3 \Rightarrow b_3 = 3$; inductively, $2 \cdot (i-1) = (i-2) + b_i \Rightarrow b_i = i$. But, then $b_{2n-1} = 2 \cdot b_{2n} \Rightarrow 2n - 1 - 4n = 0 \Rightarrow n = -\frac{1}{2}$, which is absurd. Analogously, if $\lambda = -2$, $b_i = i(-1)^{i+1}$, so we get $n = \frac{1}{6}$.

Now, we solve the recursion relation $b_k - \lambda b_{k-1} + b_{k-2} = 0$. The corresponding polynomial equation $x^2 - \lambda x + 1 = 0$ has roots

$$x_1 = \frac{\lambda + \sqrt{\lambda^2 - 4}}{2}, x_2 = \frac{\lambda - \sqrt{\lambda^2 - 4}}{2}.$$

Consequently, the general solution to the recurrence is $b_k = c_1 x_1^k + c_2 x_2^k$, and in particular $c_1 x_1 + c_2 x_2 = 1$, $c_1 x_1^2 + c_2 x_2^2 = \lambda$. Writing these out, we get:

$$(c_1 + c_2)\lambda + (c_1 - c_2)\sqrt{\lambda^2 - 4} = 2$$

$$c_1(\lambda^2 + 2\lambda\sqrt{\lambda^2 - 4} + \lambda^2 - 4) + c_2(\lambda^2 - 2\lambda\sqrt{\lambda^2 - 4} + \lambda^2 - 4) = 4\lambda$$
$$\Rightarrow 2\lambda\left((c_1 + c_2)\lambda + (c_1 - c_2)\sqrt{\lambda^2 - 4}\right) - 4(c_1 + c_2) = 4\lambda$$

Simplifying, we get $c_1 = -c_2 = -c$. We must also have $\lambda c(x_1^{2n} - x_2^{2n}) = \lambda b_{2n} = b_{2n+1} = c(x_1^{2n-1} - x_2^{2n-1})$; rewriting, we have $x_1^{2n-1}(\lambda x_1 - 1) - x_2^{2n-1}(\lambda x_2 - 1) = 0$.

Recall that $x_1 + x_2 = \lambda$, and $x_1 x_2 = 1$. We can then write $\lambda x_1 - 1 = (x_1 + x_2)x_1 - 1 = x_1^2 + x_1 x_2 - 1 = x_1^2$, and the same for x_2. The final constraint is $x_1^{2n+1} + x_2^{2n+1} = 0$, which is equivalent to $x_2 = \gamma x_1$, where γ is a $2n+1$st root of unity (excluding 1, which would lead to $x_1 = x_2 = 0$, which is impossible). Also, $x_1 x_2 = x_1^2 \gamma = 1 \Rightarrow x_1^2 = \gamma^{2n}$. So, $x_1 = \pm\gamma^n$ and $x_2 = \pm\gamma^{n+1}$. This gives us all the eigenvalues: $\lambda = \pm(\gamma + \overline{\gamma}) = 2\operatorname{Re}\gamma = \pm 2\cos(\frac{\pi k}{2n+1})$, for $k = 1..n$. Thus, the lowest positive eigenvalue of a $2n$-path is then $2\cos(\frac{\pi n}{2n+1})$.

By a more tedious calculation, we get that the lowest eigenvalue for the forked path with $2n+1$ vertices is $2\cos(\frac{\pi(2n-1)}{4n})$.

As we remarked in Conjecture 5.5, it is possible to inductively multiply out the $(x - \lambda_1)\cdots(x - \lambda_n)$ to get $\sum_{k=0}^{k}(-1)^{k-i+2}\binom{2k-i}{2(k-i)}x^{2(k-i)}$ as the characteristic polynomial.

6. Future Directions

To recapitulate, it would be exciting to prove the conjectured improvement to the upper bound on M_{2k}, or, better yet, to prove an upper bound that does not overestimate by an exponential factor. Ideally, we would like a result that allows us to match [5] for the tail probabilities. The lowest positive eigenvalue conjectures from §5 are interesting as well, while Conjectures 5.4 and 5.5 don't seem to have much independent interest.

Very little is known (rigorously) about the spectrum of the giant component; our numerical results indicate that it has a discrete component. Is this a strange artifact, or do they exist? Are they δ-functions? If they exist, is it possible to prove Erdös-Rényi style theorems about the parts of the component that originate them? Can we find the total weight on these atoms? If we subtract the discrete spectrum (from trees and from the giant component, if any), is the remainder absolutely continuos with respect to the Lebesgue measure (numerically, it seems quite smooth and well behaved)? Is it possible to find an analytic expression for this part of the spectrum?

There are also a few interesting questions that don't directly relate to this paper. Is the discrete spectrum dense in the real line? Does it cover all algebraic numbers? Do eigenvalues of arbitrary finite graphs cover all algebraic numbers?

7. Acknowledgements

I would like to thank Professor Yakov Sinai, my adviser, for his support, for his direction and insightful questions. I am grateful to Professor Benjamin Sudakov for productive discussions, and for teaching me much intuition behind graph theory. Finally, I thank Lior Silberman for bearing my many stupid questions, for his great advice, and for teaching me many useful facts.

Appendix A. Implementation and text on CD-ROM

Please see the file README on the included CD-ROM. The file documents the programs, and provides instructions for compiling them on a UNIX-like operating system. The CD-ROM also contains an electronic (PDF, DVI, and PostScript) version of the paper. The author will be available for comments and questions at `aspirido@alumni.princeton.edu` (starting September, 2004).

References

[1] P. Erdös, A. Rényi, *On the evolution of random graphs*, *On the strength of connectedness of a random graph.* As published in "Paul Erdos: The Art of Counting: Selected Writings", The MIT Press, Cambridge, Massachusets. pp. 569-624, 1973. (papers received in 1959 and 1960, respectively)

[2] A. D. Barbour, *Poisson convergence and random graphs,* Math. Proc. Camb. Phil. Soc. (1982), **92**, pp. 349-359

[3] A. Khorunzhy and V. Vengerovsky, *On asymptotic solvability of random graph's Laplacians.* preprint, available at `http://arxiv.org/abs/math-ph/0009028`.

[4] R. L. Graham, D. E. Knuth, O. Patashnik, Concrete Mathematics, Addison-Wesley, 2nd ed., p. 493.

[5] G. J. Rodgers and A. J. Bray *Density of states of a sparse random matrix*, Phys. Rev. B **37**, 3557-3562 (1988). This has a nice confirmation of our asymptotic decay estimation, albeit physical.

[6] O. Khorunzhiy (same as A. Khorunzhy) *Rooted trees and moments of large sparse random matrices,* Discrete Mathematics and Theoretical Computer Science **AC**, 2003, pp. 145-154 Available at `http://dmtcs.loria.fr/proceedings/html/pdfpapers/dmAC0114.pdf`

[7] B. Bollobás, *Random Graphs.* Academic Press, London (1985).

[8] R. Otter, *The number of trees*, Annals of Math. **49** (1948) 583-599; MR 10,53c.

[9] G. Li and F. Ruskey, *The Advantages of Forward Thinking in Generating Rooted and Free Trees*, presented at the 10th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), (1999) S939-940; available online at `http://www.cs.uvic.ca/~ruskey/Publications/RootedFreeTree.html`

[10] N. J. A.. Sloane's *On-line Encyclopedia of Integer Sequences*, available at `http://www.research.att.com/~njas/sequences/index.html`

[11] M. Krivelevich, B. Sudakov, *The largest eigenvalue of sparse random graphs*, Combinatorics, Probability and Computing (2003) **12**, pp. 61-72. Available at `http://www.math.princeton.edu/~bsudakov/sparse-eigen.pdf`.

[12] M. Bauer, O. Golinelli *On the kernel of tree incidences matrices*, Journal of Integer Sequences, **3** (200), Article 00.1.4. Available at `http://www.math.uwaterloo.ca/JIS/VOL3/BAUER/zerotree.pdf`

[13] G. Semerjian, L. F. Cugliandolo *Sparse random matrices: the eigenvalue spectrum revisited*, J. Phys. A: Math. Gen. **35** (14 June 2002) pp. 4837-4851