MOMENT BASED ESTIMATION OF STOCHASTIC KRONECKER GRAPH PARAMETERS

By

Art B. Owen

Technical Report No. 2010-14
December 2010

Department of Statistics
STANFORD UNIVERSITY
Stanford, California 94305-4065

MOMENT BASED ESTIMATION OF STOCHASTIC KRONECKER GRAPH PARAMETERS

By

Art B. Owen
Stanford University

Technical Report No. 2010-14
December 2010

This research was supported in part by National Science Foundation grant DMS 0906056.

Department of Statistics
STANFORD UNIVERSITY
Stanford, California 94305-4065

http://statistics.stanford.edu

Moment based estimation of stochastic Kronecker graph parameters

Art B. Owen
Stanford University

Orig: September 2008
This: December 2010

Abstract

Stochastic Kronecker graphs supply a parsimonious model for large sparse real world graphs. They can specify the distribution of a large random graph using only three or four parameters. Those parameters have however proved difficult to choose in specific applications. This article looks at method of moments estimators that are computationally much simpler than maximum likelihood.

1 Introduction

Stochastic Kronecker graphs were introduced by [2] as a method for simulating very large random graphs. Random synthetic graphs are used to test graph algorithms and to understand oberved properties of graphs. By using simulated graphs, instead of real measured ones, it is possible to test algorithms on graphs larger or denser than presently observed ones. Simulated graphs also allow one to judge which features of a real graph are likely to hold in other similar graphs and which are idiosyncratic to the given data. Stochastic Kronecker graphs are able to serve these purposes through a model that has only three or four parameters.

Given a node set $\mathcal{N}$ of cardinality $N \geq 1$, and a matrix $P_{ij} \in [0,1]$ defined over $i, j \in \mathcal{N}$, a random graph $G^*(P)$ is one where the edge $[ij]$ exists with probability P_{ij} and all N^2 edges exist or don't independently. The graph G^* includes loops $[ii]$ and may possibly include both $[ij]$ and $[ji]$. We snip these out by defining the random graph $G(P)$ with edges $[ij]$ only when $i \neq j$ and $[\max(i,j), \min(i,j)] \in G^*$, using any non-random ordering of $\mathcal{N}$. Both G^* and G are in fact probability weighted ensembles of graphs, but for simplicity we describe them as single random graphs. Without loss of generality we take P to be a symmetric matrix.

The description of P allows up to $N(N-1)/2$ parameters that affect the outcome. Much more parsimonious descriptions can be made by taking P to be the Kronecker product of two or more smaller matrices. Recall that the

Kronecker product of matrices $X \in \mathbb{R}^{m\times n}$ and $Y \in \mathbb{R}^{r\times s}$ is

$$X \otimes Y \equiv \begin{pmatrix} X_{11}Y & X_{12}Y & \cdots & X_{1n}Y \\ X_{21}Y & X_{22}Y & \cdots & X_{2n}Y \\ \vdots & \vdots & \ddots & \vdots \\ X_{m1}Y & X_{m2}Y & \cdots & X_{mn}Y \end{pmatrix} \in \mathbb{R}^{mr\times ns}.$$

An extremely parsimonious stochastic Kronecker graph takes P to be the r-fold Kronecker product of $\Theta = \begin{pmatrix} a & b \\ b & c \end{pmatrix}$, for $a, b, c \in [0, 1]$. That is

$$P = P^{(r)} = \Theta \otimes \Theta \otimes \cdots \otimes \Theta \equiv \Theta^{[r]}.$$

If the power r is known, then only three numbers need to be specified, and with them we can then simulate other graphs that are like the original. Perhaps surprisingly, stochastic Kronecker graphs imitate many, but of course not all, of the important features seen in large real world graphs. See for example [3].

We would like to pick parameters $a, b, c \in [0, 1]$ so as to match the properties seen in a real and large graph. First, there is clearly no way to distinguish between $\Theta = \begin{pmatrix} a & b \\ b & c \end{pmatrix}$ and $\Theta^* = \begin{pmatrix} c & b \\ b & a \end{pmatrix}$ because they give rise to the same graph distribution. To force identifiability, we may simply assume that $a \geq c$ without loss of generality.

Unfortunately, maximum likelihood estimationsof a, b, and c is computationally very awkward. The main problem is that the likelihood involves $N!$ permutations in which to order the nodes. Even for a given permutation, [3] find that it takes more than $O(N^2)$ work to evaluate the likelihood contribution. In practice many thousands or millions of randomly sampled permutations are used to estimate the likelihood.

The Kronecker structure in P makes certain aspects of G very tractable. For example the number E of edges in G can be shown to have expectation

$$\mathbb{E}(E) = \frac{1}{2}\Big((a + 2b + c)^r - (a + c)^r\Big). \tag{1}$$

Thus simply counting the edges in G gives us valuable information on the parameter vector (a, b, c). Because E is a sum of independent Bernoulli random variables we find that $\mathbb{V}\mathrm{ar}(E) \leq \mathbb{E}(E)$ and so the relative uncertainty $\sqrt{\mathbb{V}\mathrm{ar}(E)}/\mathbb{E}(E) \leq \mathbb{E}(E)^{-1/2}$ will be small in a graph with a large number of expected edges.

The outline of this paper is as follows. Section 2 derives formula (1) and several others like it. The features to be counted are illustrated in Figure 1. The expected number of triangles leads to a simpler expression in terms of a, b, and c than we get for tripins. But the latter are easier to count than triangles are in a large graph.

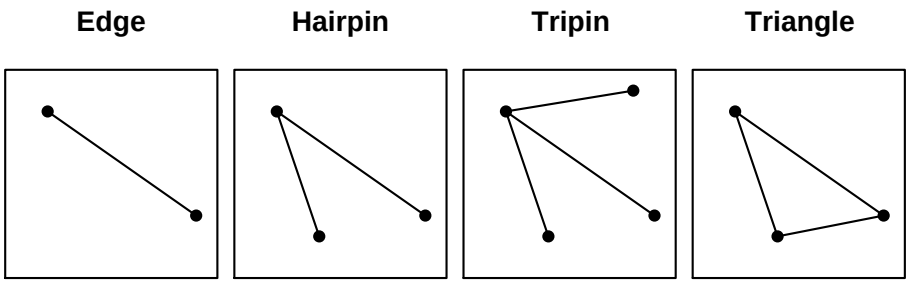

Figure 1: This figure illustrates some of the graph features that we can count, for use in moment based estimates of the parameters in the stochastic Kronecker graph.

2 Moment formulas

This section derives some formulas for the expected number of features of various types. The expected feature counts require sums over various sets of nodes. Section 2.1 records some summation formulas that simplify this task. Then Section 2.2 turns expected feature counts into sums and Section 2.3 shows how those sums simplify for stochastic Kronecker matrices.

2.1 Summation formulas

Let $i, j, k, l \in \mathcal{N}$ for a finite index set $\mathcal{N}$. A plain summation sign $\sum$ represents sums over all combinations of levels of all the subscripting indices used. The symbol $\sum^*$ includes all levels of all indices, except for any combinations where two or more of those indices take the same value. In several places we find that sums are easier to do over all levels of all indices, while the desired sums are over unique levels. Here we record some formulas to translate the second type into the first.

It is elementary that

$$\sum_{ij}^{*} f_{ij} = \sum_{ij} f_{ij} - \sum_{i} f_{ii}, \tag{2}$$

and similarly

$$\sum_{ijk}^{*} f_{ijk} = \sum_{ijk} f_{ijk} - \sum_{ij} \left(f_{ijj} + f_{iji} + f_{iij}\right) + 2\sum_{i} f_{iii}. \tag{3}$$

When there are four indices, we get

$$\begin{aligned}\sum_{ijkl}^{*} f_{ijkl} = {} & \sum_{ijkl} f_{ijkl} - \sum_{ijk}\Big(f_{ijki} + f_{ijkj} + f_{ijkk} + f_{ijik} + f_{ijjk} + f_{iijk}\Big) \\ & + \sum_{ij}\Big(2\left(f_{ijjj} + f_{ijii} + f_{iiji} + f_{iiij}\right) + f_{ijij} + f_{ijji} + f_{iijj}\Big) - 6\sum_{i} f_{iiii}.\end{aligned} \tag{4}$$

Equation (4) is more complicated than the others. It can be proved by expanding $\sum_{ijkl}^{*} f_{ijkl} = \sum_{ijk}^{*} g_{ijk}$ using (3), where $g_{ijk} = \sum_l f_{ijkl} - f_{ijki} - f_{ijkj} - f_{ijkk}$.

In some applications the first index is singled out but the others are exchangeable. In such cases $f_{ijk} = f_{ikj}$, when there are three indices, while $f_{ijkl} = f_{ijlk} = f_{ikjl} = f_{iklj} = f_{iljk} = f_{ilkj}$ is the version for four indices.

When indices after the first are exchangeable, then equation (3) simplifies to

$$\sum_{ijk} f_{ijk} - \sum_{ij} \Big(f_{ijj} + 2 f_{iij} \Big) + 2 \sum_{i} f_{iii}, \tag{5}$$

and equation (4) simplifies to

$$\sum_{ijkl} f_{ijkl} - 3 \sum_{ijk} \Big(f_{iijk} + f_{ijjk} \Big) + \sum_{ij} \Big(2 f_{ijjj} + 5 f_{iijj} + 4 f_{iiij} \Big) - 6 \sum_{i} f_{iiii}. \tag{6}$$

When all indices ijk are exchangeable, so that $f_{ijk} = f_{ikj} = f_{jik} = f_{jki} = f_{kij} = f_{kji}$, then equation (5) simplifies to

$$\sum_{ijk} f_{ijk} - 3 \sum_{ij} f_{iij} + 2 \sum_{i} f_{iii}. \tag{7}$$

2.2 Expected feature counts

The graph features we describe are shown in Figure 1. In addition to edges, there are hairpins where two edges share a common node, tripins where three edges share a node, and triangles.

Recall that G^* is a random graph with $\Pr([ij] \in G^*) = P_{ij}$ (independently). Let it have incidence matrix A^*. There may be loops $A^*_{ii} = 1$ and for $i \neq j$ A^*_{ij} and A^*_{ji} are independently generated. The graph G is formed by deleting loops from G^* and symmetrizing the incidence matrix via

$$A_{ij} = \begin{cases} A^*_{ij} & i > j \\ 0 & i = j \\ A^*_{ji} & i < j. \end{cases}$$

The number of edges in G is $E = (1/2) \sum_{ij}^{*} A_{ij}$. The expected number of edges satisfies

$$2\,\mathbb{E}(E) = \mathbb{E}\Big(\sum_{ij}^{*} A_{ij} \Big) = \sum_{ij}^{*} P_{ij} = \sum_{ij} P_{ij} - \sum_{i} P_{ii}, \tag{8}$$

using $\mathbb{E}(A_{ij}) = \mathbb{E}(A^*_{ij})$.

The number of hairpins in G is $H = (1/2) \sum_{ijk}^{*} A_{ij} A_{ik}$. The factor of two adjusts the sum for counting $\{[ij], [ik]\}$ twice. The expected value of H satisfies

$$2\,\mathbb{E}(H) = \sum_{ijk^*} P_{ij} P_{ik} = \sum_{ijk} P_{ij} P_{ik} - \sum_{ij} P_{ij}^2 - 2 \sum_{ij} P_{ii} P_{ij} + 2 \sum_{i} P_{ii}^2$$

after applying equation (3).

The number of triangles in G is $\Delta = (1/6)\sum^*_{ijk} A_{ij}A_{ik}A_{jk}$, because the sum counts each triangle $3! = 6$ times. The expected value of each term is $f_{ijk} = P_{ij}P_{ik}P_{jk}$ which is symmetric in its three arguments and so we may apply equation (7) to get

$$6\,\mathbb{E}(\Delta) = \sum_{ijk} P_{ij}P_{ik}P_{jk} - 3\sum_{ij} P_{ii}P_{ij}^2 + 2\sum_i P_{ii}^3.$$

The number of tripins in G is $T = (1/6)\sum^*_{ijkl} A_{ij}A_{ik}A_{il}$. The final three indices in $f_{ijkl} = P_{ij}P_{ik}P_{il}$ are exchangeable, and so equation (6) applies. Thus

$$\begin{aligned} 6\,\mathbb{E}(T) = &\sum_{ijkl} P_{ij}P_{ik}P_{il} - 3\sum_{ijk} P_{ii}P_{ij}P_{ik} - 3\sum_{ijk} P_{ij}^2P_{ik} \\ &+ 2\sum_{ij} P_{ij}^3 + 5\sum_{ij} P_{ii}P_{ij}^2 + 4\sum_{ij} P_{ii}^2P_{ij} - 6\sum_i P_{ii}^3. \end{aligned}$$

2.3 Simplifying the sums

The sums in the expected counts simplify, because of the properties of the Kronecker graph. Let the node set be $\mathcal{N} = \mathcal{N}_r = \{0, 1, \ldots, 2^r - 1\}$. For $i \in \mathcal{N}$ write $i = \sum_{s=1}^r 2^{s-1} i_s$ for $i_s \in \{0, 1\}$. Similarly let j, k, and l be described in terms of $j_s, k_s, l_s \in \{0, 1\}$ for $s = 1, \ldots, r$.

The matrix entry $P_{ij} = P_{ij}^{(r)}$ may be written

$$P_{ij}^{(r)} = \prod_{s=1}^r \Theta_{i_s j_s}.$$

For $r \geq 2$, we simplify the expression by induction using a smaller version of the problem defined via $P^{(r-1)}$. Specifically,

$$\begin{aligned} \sum_{ijk} P_{ij}^{(r)} P_{ik}^{(r)} &= \sum_{i_1} \cdots \sum_{i_r} \sum_{j_1} \cdots \sum_{j_r} \sum_{k_1} \cdots \sum_{k_r} \prod_{s=1}^r \Theta_{i_s j_s} \Theta_{i_s k_s} \\ &= \left(\sum_{i_1} \cdots \sum_{i_{r-1}} \sum_{j_1} \cdots \sum_{j_{r-1}} \sum_{k_1} \cdots \sum_{k_{r-1}} \prod_{s=1}^{r-1} \Theta_{i_s j_s} \Theta_{i_s k_s} \right) \sum_{i_r j_r k_r} \Theta_{i_r j_r} \Theta_{i_r k_r} \\ &= \left(\sum_{ijk} P_{ij}^{(r-1)} P_{ik}^{(r-1)} \right) \sum_{i_r j_r k_r} \Theta_{i_r j_r} \Theta_{i_r k_r} \\ &= \left(\sum_{i_r j_r k_r} \Theta_{i_r j_r} \Theta_{i_r k_r} \right)^r, \end{aligned}$$

where indices i_s, j_s and k_s are summed over their full ranges, and the indices i, j, k for $P_{ij}^{(r-1)} P_{ik}^{(r-1)}$ are summed over the node set $\mathcal{N}_{r-1} = \{0, \ldots, 2^{r-1} - 1\}$.

i	j	k	Θ_{ii}	Θ_{ij}	Θ_{ik}	Θ_{jk}
0	0	0	a	a	a	a
1	0	0	c	b	b	a
0	1	0	a	b	a	b
1	1	0	c	c	b	b
0	0	1	a	a	b	b
1	0	1	c	b	c	b
0	1	1	a	b	b	c
1	1	1	c	c	c	c

Table 1: This table shows entries in the matrix Θ with various indexing patterns needed in the examples. Sums over i, ij, and ijk use, respectively, the first 2, 4, and 8 rows of the table.

All of the sums of products of elements of $P_{ij}^{(r)}$ listed in the previous section also reduce this way to r'th powers of their value for the case $r = 1$.

Continuing, and using $\Theta = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$ one finds via Table 1 that

$$\sum_{i_r j_r k_r} \Theta_{i_r j_r} \Theta_{i_r k_r} = a^2 + ba + b^2 + cb + ab + b^2 + bc + c^2,$$

so that

$$\sum_{ijk} P_{ij} P_{ik} = \big((a+b)^2 + (b+c)^2\big)^r.$$

In the rest of this section, we record the sums we need. First, the sums over one index variable:

$$\sum_{i} P_{ii}^m = \big(a^m + c^m\big)^r, \tag{9}$$

where cases $m = 1, 2, 3$ are used in the counts. Next, sums over two index variables:

$$\sum_{ij} P_{ii}^m P_{ij}^n = \big(a^m(a^n + b^n) + c^m(b^n + c^n)\big)^r,$$

where the cases we need are for $(m, n) \in \{(0,1), (0,2), (0,3), (1,1), (1,2)\}$.

Four sums over three indices are used. They are:

$$\sum_{ijk} P_{ij} P_{ik} = \big((a+b)^2 + (b+c)^2\big)^r$$

$$\sum_{ijk} P_{ij}^2 P_{ik} = \big(a^3 + c^3 + b(a^2 + c^2) + b^2(a+c) + 2b^3\big)^r$$

$$\sum_{ijk} P_{ij} P_{ik} P_{jk} = \big(a^3 + c^3 + 3b^2(a+c)\big)^r \quad \text{and}$$

$$\sum_{ijk} P_{ii} P_{ij} P_{ik} = \big(a(a+b)^2 + c(b+c)^2\big)^r.$$

Finally, one sum over four indices is used:

$$\sum_{ijkl} P_{ij}P_{ik}P_{il} = \big((a+b)^3 + (b+c)^3\big)^r.$$

2.4 Specific formulas

Gathering together the previous developments, we find

$$\begin{aligned}
2\,\mathbb{E}(E) &= \big(a+2b+c\big)^r - \big(a+c\big)^r\\
2\,\mathbb{E}(H) &= \big((a+b)^2 + (b+c)^2\big)^r - 2\big(a(a+b)+c(c+b)\big)^r\\
&\quad - \big(a^2+2b^2+c^2\big)^r + 2\big(a^2+c^2\big)^r\\
6\,\mathbb{E}(\Delta) &= \big(a^3+3b^2(a+c)+c^3\big)^r - 3\big(a(a^2+b^2)+c(b^2+c^2)\big)^r + 2\big(a^3+c^3\big)^r\\
6\,\mathbb{E}(T) &= \big((a+b)^3+(b+c)^3\big)^r - 3\big(a(a+b)^2+c(b+c)^2\big)^r\\
&\quad - 3\big(a^3+c^3+b(a^2+c^2)+b^2(a+c)+2b^3\big)^r + 2\big(a^3+2b^3+c^3\big)^r\\
&\quad + 5\big(a^3+c^3+b^2(a+c)\big)^r + 4\big(a^3+c^3+b(a^2+c^2)\big)^r - 6\big(a^3+c^3\big)^r.
\end{aligned}$$

In each formula, the terms from sums over fewer indices come after the ones from more indices. For large r those earlier terms are more important. For example the first term in $\mathbb{E}(E)$ is $(1+2b/(a+c))^r$ times as large as the second one, which subtracts out loops. The first term will dominate for large r unless $b \ll a+c$. The relative magnitude of the second term is

$$\left(\frac{a+c}{a+2b+c}\right)^r = 2^{r\log_2((a+c)/(a+2b+c))} = N^{-\alpha}$$

where $\alpha = \log_2((a+2b+c)/(a+c))$. If $\alpha > 1/2$ then dropping the second term in $\mathbb{E}(E)$ makes a smaller difference than the sampling uncertainty in E. This holds when the off diagonal element of Θ is not too small compared to the average of the diagonal elements: $b > (\sqrt{2}-1)(a+c)/2$.

2.5 Test cases

Some special cases are useful as checks on these formulas. For example if $b = 0$ then there are no edges in G^* apart from loops. As a result G has 2^r isolated nodes. We find from the above that $\mathbb{E}(E) = \mathbb{E}(H) = \mathbb{E}(\Delta) = \mathbb{E}(T) = 0$ when $b = 0$.

If instead $a = c = 0$ then each node $i \in \mathcal{N}$ with coordinates $i_1, \dots, i_r$ has a dual node i^* which has coordinates $i^*_s = 1 - i_s$ for $s = 1, \dots, r$. The only possible edges in G are between nodes and their duals. There are $N = 2^r$ nodes each with probability b^r of having an edge out to its dual. The formula above correctly gives $\mathbb{E}(E) = (2b)^r/2 = Nb^r/2$ when $a = c = 0$. We also get $\mathbb{E}(H) = \mathbb{E}(\Delta) = \mathbb{E}(T) = 0$ when $a = c = 0$.

If $a = b = c = 1$, then G^* has every possible edge and loop with probability 1. As a result G is the complete graph on $N = 2^r$ nodes. Then it has $N(N-1)/2$ edges, $N(N-1)(N-2)/2$ hairpins, $N(N-1)(N-2)/2$ triangles, and it has $N(N-1)(N-2)(N-3)/6$ tripins.

2.6 Numerical example

The PageRank derby [1], used stochastic Kronecker graphs to illustrate some algorithms. They present two examples with $r = 14$ so $N = 2^{14} = 16{,}384$. They used matrices $\Theta_g \propto \begin{pmatrix} 0.45 & 0.15 \\ 0.15 & 0.25 \end{pmatrix}$, for what they call their nice case, and $\Theta_b \propto \begin{pmatrix} 0.57 & 0.19 \\ 0.19 & 0.05 \end{pmatrix}$ for the nasty case. The entries in these matrices sum to one, so they describe a setting where $\mathbb{E}(E^*) = 1$. In fact each example had 131,072 edges in the graph G^* including loops and possible duplicate entries. Thus the parameters should satisfy $(a + 2b + c) \doteq 131{,}072^{1/14} \doteq 2.320$.

The number of triangles in G after loops are deleted and the incidence matrix is symmetrized, should therefore be about $(a^3 + 3b^2(a + c) + c^3)^{14}/6$ which is roughly 1600 for the nice case and 1.6×10^6 for the nasty case.

3 Solving for a, b, and c

There are four equations in Section 2.4. To estimate a, b, and c will require at least three of them. Because they are high order polynomials it is possible that there are multiple solutions or even none at all. The latter circumstance would provide some evidence of lack of fit of the stochastic Kronecker model to a given graph.

A pragmatic way to choose a, b, and c is to solve

$$\min_{a,b,c} \sum_{F} \frac{(F - \mathbb{E}_{a,b,c}(F))^2}{\mathbb{E}_{a,b,c}(F)} \tag{10}$$

where the sum is over three or four of the features $F \in \{E, H, T, \Delta\}$ from Section 2.4 and the minimization is taken over $0 \le c \le a \le 1$ and $0 \le b \le 1$. The terms in (10) are scaled by an approximate variance. A sharper expression would account for correlations among the features used. That would increase statistical efficiency, but in large problems lack of fit to the Kronecker model is likely to be more important than inefficiency of estimates within it.

Because there are only three parameters the criterion (10) can simply be evaluated over a grid inside $\{(a, b, c) \in [0, 1]^3 \mid a \ge c\}$. To be sure of taking a point within ε of the minimizer takes work $O(\varepsilon^{-3})$. This section looks at ways to reduce that effort.

3.1 Counting features in a graph

Three of the features we use are easily obtainable from the degrees of the nodes. Let $d_i = \sum_{j \in \mathcal{N}} A_{ij}$ be the degree of node i in graph G. Then

$$\begin{aligned} E &= \frac{1}{2} \sum_i d_i, \\ H &= \frac{1}{2} \sum_i d_i(d_i - 1), \quad \text{and} \\ T &= \frac{1}{6} \sum_i d_i(d_i - 1)(d_i - 2) \end{aligned}$$

give the number of edges, hairpins, and tripins in terms of the degrees d_i.

The number of triangles Δ is not a simple function of d_i. Algorithms to count triangles are considered in [5]. The time complexity can be as low as $O(E^{3/2})$.

3.2 Moments

In a synthetic graph $N = 2^r$ is known. When fitting to a real world graph a pragmatic choice is $r = \log_2(\lceil N \rceil)$. The interpretation is that the random graph G^* may have had isolated nodes that were then dropped when forming G, but we suppose that fewer than half of the nodes in G^* have been dropped.

If we consider just the lead terms, then we could get estimates $\hat{a}$, $\hat{b}$, and $\hat{c}$ by solving three of the equations:

$$\begin{aligned} e \equiv (2E)^{1/r} &= \hat{a} + 2\hat{b} + \hat{c}, \\ h \equiv (2H)^{1/r} &= (\hat{a} + \hat{b})^2 + (\hat{b} + \hat{c})^2, \\ \delta \equiv (6\Delta)^{1/r} &= (\hat{a}^3 + \hat{c}^3) + 3\hat{b}^2(\hat{a} + \hat{c}) \quad \text{and} \\ t \equiv (6T)^{1/r} &= (\hat{a} + \hat{b})^3 + (\hat{a} + \hat{c})^3. \end{aligned}$$

The equations for e and h together can be solved to get

$$\begin{aligned} \hat{x} \equiv \hat{a} + \hat{b} &= \frac{e + \sqrt{2h - e^2}}{2} \\ \hat{y} \equiv \hat{b} + \hat{c} &= \frac{e - \sqrt{2h - e^2}}{2}, \end{aligned} \tag{11}$$

where we have assumed that $a \geq c$. The transformed tripin count t matches $\hat{x}^3 + \hat{y}^3$ and so it is redundant given e and h, if we are just using lead terms. We must either count triangles, or use higher order terms.

Equation (11) may fail to have a meaningful solution. At a minumum we require $e^2 \leq 2h$ and $e \geq \sqrt{2h - e^2}$. These translate into

$$h \leq e^2 \leq 2h,$$

which is equivalent to

$$2H \le 4E^2 \le 2^{r+1}H$$

that in terms of node degrees is

$$\sum_i d_i(d_i-1) \le \Big(\sum_i d_i\Big)^2 \le N\sum_i d_i(d_i-1). \tag{12}$$

The left hand inequality in (12) holds for any graph, but the right hand side need not. It holds when $N^{-1}\sum_i (d_i-\bar{d})^2 \ge \bar{d} = N^{-1}\sum_i d_i$. If the variance of the node degrees d_i is smaller than their mean, then equation (11) does not have real valued solutions. The degree distribution of a stochastic Kronecker graph has heavy tails [4]. Therefore in applications where that model is suitable equation (11) will give a reasonable solution.

When d_i have a variance larger than their mean, then we can do a univariate grid search for $b \in [0,1]$ using equation (11) to get $a = x - b \equiv a(b)$ and $c = y - b \equiv c(b)$. The choice of b can then be made as the minimizer of $|a(b)^3 + c(b)^3 + 3b^2(a(b)+c(b)) - \delta|$.

4 Applications

Since writing this I have heard from David Gleich who has tried out the methods on some graphs. The fitted values seem to miss the number of triangles in the real graphs. This may be because Kronecker models do not produce quite as many triangles as real graphs have.

References

[1] K. Devine, J. Berry, and S. Plimpton. The PageRank derby. Technical Report SAND2008-5162P, Sandia National Laboratories, 2008.

[2] J. Leskovec, D. Chakrabarti, J. Kleinberg, and C. Faloutsos. Realistic, mathematically tractable graph generation and evolution, using Kronecker multiplication. In *European Conference on Principles and Practice of Knowledge Discovery in Databases (UCML/PKDD)*, 2005.

[3] J. Leskovec and C. Faloutsos. Scalable modeling of real graphs using Kronecker multiplication. In *International Convference on Machine Learning (ICML)*, 2007.

[4] M. Mahdian and Y. Xu. Stochastic Kronecker graphs. In *Proceedings of the 5th Workshop on Algorithms and Models for the Web-graph (WAW2007)*, pages 179–186, 2007.

[5] T. Schank and D. Wagner. Finding, counting, and listing all triangles in large graphs. In *Workshop on Experimental and Efficient Algorithms (WEA)*, 2005.

Appendix: sage code for formulas

```
var('a,b,c,r,N') ; N = 2^r

def sum_i_Piim(a,b,c,m,r):
  return (a^m+c^m)^r

def sum_ij_Piim_Pijn(a,b,c,m,n,r):
  return (a^m*(a^n+b^n)+c^m*(b^n+c^n))^r

def sum_ijk_Pij_Pik(a,b,c,r):
  return ((a+b)^2+(b+c)^2)^r

def sum_ijk_Pij_Pik_Pjk(a,b,c,r):
  return (3*b^2*(a+c)+a^3+c^3)^r

def sum_ijk_Pii_Pij_Pik(a,b,c,r):
  return (a*(a+b)^2+c*(b+c)^2)^r

def sum_ijk_Pij2_Pik(a,b,c,r):
  return (a^3+c^3+b*(a^2+c^2)+b^2*(a+c)+2*b^3)^r

def sum_ijkl_Pij_Pik_Pil(a,b,c,r):
  return ((a+b)^3+(b+c)^3)^r
##
## Expected number of edges in G^*
##
def estar(a,b,c,r):
  return sum_ij_Piim_Pijn(a,b,c,0,1,r)-sum_i_Piim(a,b,c,1,r)
##
## Expected number of hairpins in G^*
##
def hstar(a,b,c,r):
  ans  = sum_ijk_Pij_Pik(a,b,c,r)
  ans -= sum_ij_Piim_Pijn(a,b,c,0,2,r)
  ans -= 2*sum_ij_Piim_Pijn(a,b,c,1,1,r)
  ans += 2*sum_i_Piim(a,b,c,2,r)
  return ans
##
## Expected number of triangles in G^*
##
def tristar(a,b,c,r):
  ans  = sum_ijk_Pij_Pik_Pjk(a,b,c,r)
  ans -= 3*sum_ij_Piim_Pijn(a,b,c,1,2,r)
  ans += 2*sum_i_Piim(a,b,c,3,r)
  return ans
```

```
##
## Expected number of formal tripins in G^*
##
def tripstar(a,b,c,r):
  ans  = sum_ijkl_Pij_Pik_Pil(a,b,c,r)
  ans -= 3*sum_ijk_Pii_Pij_Pik(a,b,c,r)
  ans -= 3*sum_ijk_Pij2_Pik(a,b,c,r)
  ans += 2*sum_ij_Piim_Pijn(a,b,c,0,3,r)
  ans += 5*sum_ij_Piim_Pijn(a,b,c,1,2,r)
  ans += 4*sum_ij_Piim_Pijn(a,b,c,2,1,r)
  ans -= 6*sum_i_Piim(a,b,c,3,r)
  return ans
##
## Print LaTeX output ... looks better after manual cleanup
##
print "LaTeX for estar"
print latex(estar(a,b,c,r))

print "LaTeX for hstar"
print latex(hstar(a,b,c,r))

print "LaTeX for tristar"
print latex(tristar(a,b,c,r))

print "LaTeX for tripstar"
print latex(tripstar(a,b,c,r))

print "test cases with b=0"

print estar(a,0,c,r)
print hstar(a,0,c,r)
print tristar(a,0,c,r)
print tripstar(a,0,c,r)

print "test cases with a=c=0"

print estar(0,b,0,r)
print hstar(0,b,0,r)
print tristar(0,b,0,r)
print tripstar(0,b,0,r)

print "test cases with a=b=c=1"
print "these expressions are zero ... but sage does not say so"

print factor( estar(1,1,1,r)/2 - (2^r)*(2^r-1)/2 )
print factor( hstar(1,1,1,r)/2 - 2^r*(2^r-1)*(2^r-2)/2 )
```

```
print factor( tristar(1,1,1,r)/6 - 2^r*(2^r-1)*(2^r-2)/6 )
print factor( tripstar(1,1,1,r)/6 - (2^r)*(2^r-1)*(2^r-2)*(2^r-3)/6 )
```