

USING COMBINATORICA/ MATHEMATICA FOR STUDENT PROJECTS IN RANDOM GRAPH THEORY

Thomas J. Pfaff¹ and Michele Zaret²

ADDRESS: (1) Mathematics Department, Ithaca College, Ithaca NY 14850-7284 USA. tpfaff@ithaca.edu and (2) 14 Gatehouse Pl., Walden NY 12586 USA. mzaret02@yahoo.com.

ABSTRACT: We give an example of a student project that experimentally explores a topic in random graph theory. We use the *Combinatorica* package in *Mathematica* to estimate the minimum number of edges needed in a random graph to have a 50 percent chance that the graph is connected. We provide the *Mathematica* code and compare it to the known theoretical result.

KEYWORDS: *Combinatorica*, *Excel*, *Mathematica*, random graph.

1 INTRODUCTION

The *Combinatorica* package in *Mathematica* allows one to do research in discrete mathematics, especially in random graph theory. With the current speed of desktop computers, students can now investigate problems that were previously inaccessible. In this paper, we describe one such project done by a student in the hope that it will encourage further research by students in this area.

Before getting to the details it is worth mentioning how this project evolved. We started with an independent study course, that followed a course in combinatorics and graph theory, to learn about *Combinatorica* and to further explore ideas in graph theory. We used the book *Computational Discrete Mathematics: Combinatorics and Graph Theory with Mathematica* by Pemmaraju and Skiena, [2]. Based on interest we were focusing on

Chapter 7 which is on properties of graphs. In particular, it was exercise 2 in section 7.6.3,

Experiment with connectivity in random graphs. Estimate the expected connectivity of a graph on n vertices with m edges for small n and m . How many edges are usually necessary before the graph is connected?

that provided the motivation that resulted in the work described in this paper. We should note that exercises in this book are split up into three types; thought exercises, programming exercises, and experimental exercises. The above exercise is an experimental exercise. After discussions, we reformulated this exercise into the question that begins the next section, and then looked to compare our results to theoretical results, which we did by referencing the book *Random Graphics* by Bollobás [1].

2 THE PROJECT

If we randomly choose a graph with n vertices, what is the fewest number of edges needed so that the graph has at least a 50 percent chance of being connected? Equivalently, we're looking for $e(v)$ so that $P(G(v, e(v)) \text{ is connected}) \geq 0.5$, where $G(v, e(v))$ is a random graph with v vertices and $e(v)$ edges. The definition of random graph that we are using is a graph which has v vertices and $e(v)$ randomly placed edges, rather than the random graph that is created from the complete graph where an edge remains or is removed with a certain probability.

Using the package *Combinatorica*, which is built into *Mathematica*, we wrote a program that generated random graphs with a fixed number of edges to estimate the number of edges, on a fixed set of vertices, so that the graph has a 50 percent chance of being connected. The program is fully explained in the next section. For vertex sets of size 10 to 65 we ran the program in small intervals for a sample size of 72 and then we transferred the output into *Excel*. We created a 90 percent confidence interval with an error of 1 to assure us that we were off by no more than 1 edge. From the confidence interval, we estimated the number of edges needed for the graph to have a 50 percent chance of being connected, and we plotted the number of vertices against the number of edges and fit a curve to it. What one might expect would be that the number of edges needed to make the graphs connected would grow exponentially, or at least rapidly, with respect to the number of vertices. What the data show is actually quite surprising.

3 METHODS

To generate our data we used the following code:

```
For[k = 5, k < 70, k += 5;
  For[n = 1, n < 73, n++,
    For[e = k - 1, e < k!/(2!*(k - 2)!), e++; j = 0;
      For[i = 1, i < 11, i++, m = EdgeConnectivity
        [ExactRandomGraph[k, e]];
      If[m > 0, {j++},]]If[
1/2 <= j/10 <=
  1, {Print["average edgeconnectivity= ",
    j/10, " ", "edges= ", e,
    e = k!/(2!*(k - 2)!)}]]].
```

To understand the code, working from the inside out, the **ExactRandomGraph** command creates the random graph using the above definition for the given number of k vertices and e edges. The **EdgeConnectivity** command gives the minimum number of edges whose deletion from the **ExactRandomGraph** disconnects it. The entire code first creates a random graph with exactly e edges and k vertices and then checks the edge connectivity. If the edge connectivity is greater than zero, a counter j counts the graph as connected and with each connected graph, j increases by 1. Beginning with k vertices, the code starts by generating random graphs with $e = k - 1$ edges, as this is the least number of edges needed for any graph to be connected, up to $\binom{k}{2}$, as this is the number of edges in the complete graph on k vertices. It then increases the number of edges until the average of the number of connected graphs is greater than or equal to $1/2$. The data is then printed out and then the code repeats everything 72 times on the same set of vertices before increasing the vertex size by 5, until it completes the process for 65 vertices.

The sample size of 72 that we ultimately chose came from a sample we ran on 5 through 65 vertices, with $n = 10$. From this we took the largest standard deviation, which was approximately 5.14 and used the formula $n = ((z_{\alpha/2})(\sigma)/E)^2$. We initially wanted a 95 percent confidence interval that trapped only one integer, but with an error of 0.5 this resulted in a sample size of 406, which was too large for our personal computers to run the program. To make the sample size smaller we dropped the confidence interval down to 90 percent and used 1 as our error, which assures us that we're off by no more than one edge; this gave the sample size of 72. Since using an error of 1 captures two integers in the confidence interval, we decided to take the larger of the two because we wanted to have a probability

of $1/2$ that the graph was connected; this gave us reasonable certainty that we were choosing the right number of edges. We ran the program in small intervals as the larger the number of vertices, the slower the computer ran and we collected all of the data in approximately 5 days, running the program on three different computers. We then transferred all of the data into **Excel**, created the confidence interval, chose the number of edges, and plotted the number of vertices against the number of edges using a scatter plot.

4 RESULTS

After plotting the data, we fit both a power curve and a linear function to the data. See Figure 1. Both types of curves fit the data well, but the power curve fits slightly better. The equation for the regression line is $y = 2.072x - 11.951$ with $R^2 = 0.9979$ and the equation for the power curve is $y = 0.6795x^{1.2524}$ with $R^2 = 0.9997$. We also plotted the residuals to see if we could see which line was a better fit.

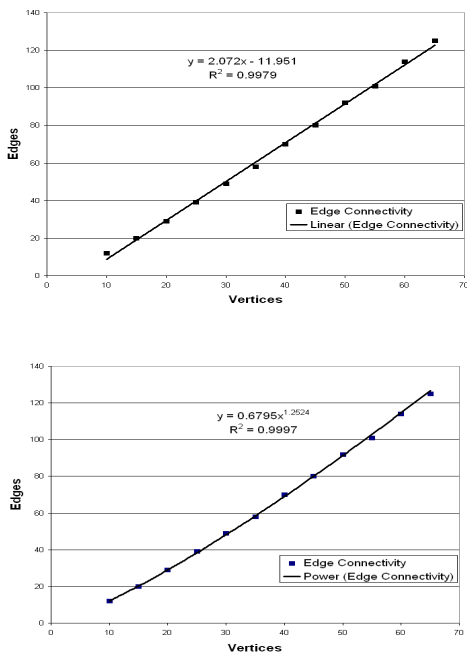


Figure 1. Fitted Line and Power Graphs

As Figure 2 shows, there is no distinct pattern in the residual plots for the power curve. On the other hand, it does appear as if the residual plot

for the linear function had an upside down U-shape. We would like to see if the curves of best fit change appreciably with a larger data set; however, a personal computer is not powerful enough to run this program (the program actually crashed a computer while it ran for 60 vertices). The conjecture that these data lead us to is quite different from what our intuition told us, since the best fit curve is nowhere near quadratic, let alone exponential.

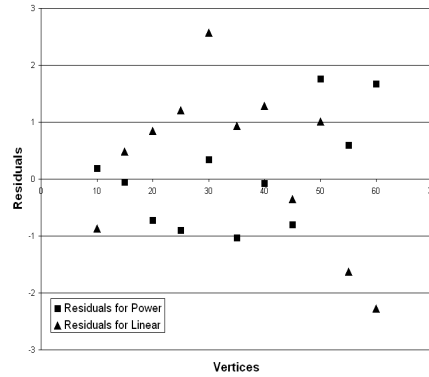


Figure 2. Residual Plots

In fact, the difficulty in deciding on which curve fits better isn't surprising. Based on Theorem 7.3 in [1], the number of vertices needed for a 50% chance of connectivity is $\frac{n}{2}(\ln n + \ln(\ln 2) + o(1))$, where $o(1)$ is a term that goes to 0 as $n \rightarrow \infty$. In other words, roughly speaking we need $n \ln n$ vertices, the growth of which is between the two functions we obtained here. So, our results here compare well to the theoretical results, and our use of *Combinatorica* allows us to gain an understanding of this theorem in an experimental fashion.

5 CONCLUSION

The theory of random graphs is not typically included in the undergraduate curriculum due to its complexity. Both books [1, 2] provide an excellent source of ideas for projects that allow students to delve into random graph theory. Many of the theorems in [1] can be investigated with the ideas here, whereas [2] has a number of good exercises that can get students started and provides the code that is needed in *Mathematica*. Finally, projects such as these are particularly nice since they combine a number of areas of mathematics (graph theory, statistics, and curve fitting) with technology use (*Mathematica* programming and *Excel*).

REFERENCES

1. Bollobás, B. 2001. *Random Graphs 2nd Ed.* Cambridge, UK: Cambridge University Press.
2. Pemmaraju, S. and S. Skiena. 2003. *Computational Discrete Mathematics: Combinatorics and Graph Theory in Mathematica.* Cambridge, UK: Cambridge University Press.

BIOGRAPHICAL SKETCH

Tom Pfaff is an associate professor of mathematics at Ithaca College, where he commutes to campus by bike all year. When not working with students on projects he is busy, along with his wife, raising their four sons. He stays in shape by rowing, running, cycling, (in fact, he has a BS in exercise science) and yes chasing kids. More about him can be found at <http://www.ithaca.edu/tpfaff>.

Michele Zaret received her BA in Mathematics in 2004 from Ithaca College. She is currently working towards her MAT in Mathematics and her MA in Mathematics at SUNY New Paltz, where she has an assistantship with the Associate Dean of Education.