1. Introduction

All too often we design procedures or algorithms that behave exceptionally when given well-formed data, but fail when applied to real-world data. Applying various clustering methods to databases of inter-linking articles is a perfect example. Real datasets are usually too sparse and possess far too few relationships between nodes to be meaningful. Specifically, traditional clustering methods are ineffective when applied to data that obeys a power-law distribution over the degree of the nodes [1].

1.1 Background

Clustering is a process that observes a set of objects (vertices) that have relationships (edges) with other objects in the set. The criterion used to define an edge can be physical distances between vertices, or even links between them. Using these edges in and among the vertices, a clustering method will group the points into communities.

These methods clearly perform well when the data is well formed. That is, if the communities are clearly established within the data, the clustering method applied will have little problem distinguishing one set of points from another. Problems arise when the data fits what is known as a power-law distribution among the degrees of the vertices. A power-law distribution indicates that a small number of points have very high degree, but the vast majority of vertices have very few edges.

A relevant example is the World Wide Web; hyperlinks between websites across the Internet as a whole also establish a relationship that obeys the power-law [2]. The prevalence of power-law distributions requires improving available clustering methods, or at least establishing the point at which the methods will no longer produce a satisfactory result.

1.2 Potential Applications

Clustering would allow for useful applications in information science if it were possible to apply the process to real data. Professors John Hopcroft and Bart Selman of the Department of Computer Science at Cornell University are trying to apply clustering methods to both the Citeseer database and the database of patents with the U.S. Patent Office, both of which show power-law distributions among the degrees of each node in the set [1].

There are numerous direct benefits of successful clustering in the Citeseer database. Each paper in the database links to other papers and is cited as a reference by several others. Each article can be placed in one or more categories. Using clustering methods, classifying a new paper can
be automated instead of requiring a librarian to read a paper and categorize it manually. This both bypasses possible misclassification and saves time. A separate but more interesting application would be determining the presence of new categories. If the topics of computational biology and spam filtering began to intersect [3] the emergence of a cutting-edge field could be perceived and examined immediately.

These benefits can also be seen when clustering is applied to the U.S. Patent Office. The process of classifying patents could become more accurate, save time [4] and could lead to the discovery of burgeoning fields. The only setback in both datasets is the presence of power-law degradation over the degree of nodes. When one paper with three links is introduced to a database of thousands of papers, it is difficult for an algorithm to emphasize the relevance of a link instead of dismissing it as noise.

1.3 Problem definition

As mentioned above, both Citeseer and U.S. Patent Office databases operate on such large sets of data, the actual inter-links between documents or patents become very sparse. Fortunately there are methods available that enhance the signal and reduce the noise within the data, specifically LaPlacian and spectral approximation methods.

Previous work has mathematically established the fact that using the LaPlacian operation with the spectral partitioning algorithm allows for perfect clustering results. The research aims to empirically illustrate that use of the LaPlacian operation has similar results with the k-means algorithm. Furthermore, we aim to establish the boundaries of the algorithms and more clearly define when the data is truly too sparse to be meaningful.

2. Procedure

2.1 Probability matrix and \( \bar{P} \)

The first step in the research is synthesizing sample data to process and experiment with. The best way to represent the sample undirected graphs is with symmetric adjacency matrices. Let \( P(i,j) = 1 \) denote that vertices i and j share an edge, and let \( P(i,j) = 0 \) signify a lack of an edge. For the sake of simplicity, our research will focus on the case of two equally sized communities.

For notation, let a matrix \( A \) denote a matrix before rounding values to 0’s and 1’s, and \( \bar{A} \) signify that the values have been rounded appropriately. Begin creating the simple matrix, \( \bar{P} \), by filling in an \( n \times n \) probability matrix \( P \) with random values between 0 and 1. Next, select two probabilities where \( q1 > q2 \). Split the matrix into four equally sized submatrices.

\[
P = \begin{bmatrix}
q1 & q2 \\
q2 & q1
\end{bmatrix}
\]
For each entry the upper left submatrix, generate an edge with probability $q_1$ (i.e. if $P(i,j) < q_1$, set $\bar{P}(i,j) = 1$ else set $\bar{P}(i,j) = 0$). Use $q_2$ in the upper right and lower left, and $q_1$ again in the bottom right. This means that the first $n/2$ vertices have a high probability ($q_1$) that they are interconnected and same for the last $n/2$ vertices. Each cluster of vertices has a low probability ($q_2$) of being connected with a vertex from the other community.

### 2.2 k-means, fitness and misclassification

For the research, we are examining clustering on datasets using the k-means algorithm for clustering. In addition, we are only dealing with the two community case, and as such will use the 2-means case. Alternative methods include spectral partitioning and agglomerative clustering. Since we are creating datasets based on a predetermined cluster arrangement, success can be defined in an algorithm’s ability to rediscover these patterns by examining synthesized data.

For example, a matrix of the form we’ve established with $P$ indicates that the first $n/2$ vertices should belong to the same cluster, as with the last $n/2$ vertices. A perfect clustering trial would result in a membership vector where the first $n/2$ elements are the same and the last $n/2$ elements are the same. Each element that is put in the wrong cluster is added to the error count. Another important measure is the misclassification of rows in the matrix. At lower $q_1$ and $q_2$ values, there may be half of a row that has so many edges that it should actually belong to a $q_1$ region instead of a $q_2$ region and vice versa.

The ratio of $q_1 : q_2$ is a factor in how accurate 2-means will be when operating on the dataset. Through preliminary trials, we’ve established that a ratio of $q_2 = \frac{1}{2} q_1$ is high enough to allow for consistent success.

### 2.3 Power law distribution

The next step is to introduce the power law distribution. Let the degree matrix $D$ be a diagonal matrix where each entry $(i,i)$ specifies the degree of vertex $i$, $d_i$. With the case of the power law, the degree information drops off rapidly. To determine these degrees, the following equation must hold:

$$n = \sum_{i=1}^{n} d_i$$

where $d_i = \frac{c}{i^3}$

The constant $c$ is a factor used to ensure that the sum of the degrees is equal to the number of vertices. The results for the case $n=1000$ show that 831 vertices should have degree 1, 104 should have degree 2, 31 should have degree 3 and so on. Introduction of the power law information to $P$ is achieved by left and right matrix multiplication by a degree matrix $D$:

$$M = D \times P \times D$$

$M$ must subsequently be rounded to 0’s and 1’s to form $\bar{M}$. After power law injection, 2-means begins to behave quite poorly, dropping to nearly 50% success rate.
2.4 The LaPlacian Operation

At this point the LaPlacian operation is used to “clean up” the data by reducing the overall variance associated with the edges in $M$. Previous work mathematically proves that using the LaPlacian would boost the signal without boosting the noise [5]. This was performed in context of spectral partitioning. Again our goal is to demonstrate that the operation would also increase success with 2-means clustering.

The LaPlacian operation is performed by the following:

$$\overline{L} = D^{-0.5} \times M \times D^{-0.5}$$

After this transformation is applied, the degree information will remain in the structure but the variance will be reduced. When the LaPlacian is applied, fitness for 2-means clustering jumps dramatically, as expected.

2.5 Spectral approximation

The last type of operation worth examining is spectral approximation to a rank 2 matrix. From $\overline{L}$, obtain the two largest magnitude eigenvalues $\lambda_1$ and $\lambda_2$, and their corresponding eigenvectors $v_1$ and $v_2$. The approximation is formed by calculating:

$$\overline{L}_s = \lambda_1 \cdot v_1 \cdot v_1^T + \lambda_2 \cdot v_2 \cdot v_2^T$$

3. Results and Open Issues

3.1 Comparison of various methods

![Figure 1a](image1.png)  
![Figure 1b](image2.png)  

Figure 1a – Results for 2-means clustering on data ranging from $q_1=0.01$ to $q_1=0.95$. Note the convergence of three methods and slight dropoff as $q_1$ approaches 1.

Figure 1b – Comparison of results for 2-means clustering on simple data, data with power law information and LaPlacian applied, and with spectral applied. Results are for low levels of $q_1$. 

Figure 1a illustrates results from experimental trials, averaged over 10 runs each and $q1$ ranging from 0.01 to 0.95. As expected, 2-means on the LaPlacian transformed matrix consistently produced higher fitnesses than 2-means on the power-law generated matrix. For low values of $q1$ (see Figure 1b), spectral performs the best and stays ahead until $q1$ is 0.02. LaPlacian joins spectral and the two methods quickly converge to a near 100% success rate. We can also see that the simple $\overrightarrow{P}$ matrix yields worse results than matrices with power law information and LaPlacian or spectral operations applied. This is most likely due to the fact that the process of adding degree information increases the probability of edges for some regions, thus adding new graph structure that helps success rates.

Notice that as the value of $q1$ increases towards 1, LaPlacian fitness slightly drops off linearly. We believe that it may be a result of having too much edge information in the graph. Since the values of $q1$ and $q2$ are relatively high, and power law information adds even more edge information to the graph, 2-means may have a difficult time discovering the two communities.

A possible fix for this problem is matrix renormalization to reduce the amount of edge information in the graph. Normalization occurs before calculating $\overrightarrow{M}$:

$$M = D \times P \times D$$
$$d = \text{diag}(D)$$

$$M = \frac{M}{d \cdot d^T}$$

Unfortunately adding normalization results in a dramatic decline in fitness, so we omitted it in our final results.

### 3.2 Larger-sized matrices

Throughout our study, we worked with matrices of size 1000 x 1000. This size was large enough to rule out any problems with fitness, yet small enough so that the scripts would run in reasonable time. With matrices of size 100 x 100 there was a greater fluctuation in success rate due chiefly to a lack of sufficient graph structure.

In general, small discrepancies are amplified at smaller scales; size $n=1000$ reduced these inconsistencies. It would however be useful to run our scripts with matrices of size 2000 and 4000, just to have empirical evidence of this.

### 3.3 Different algorithms

In our study, we worked with k-means (specifically, 2-means because we wanted to reconstruct two clusters). The performance of clustering algorithms like the agglomerative method is an open question. There are additional modifications that can be made to 2-means as well, including the use of random start points. Furthermore, the distance criterion used by the algorithm can be switched from Euclidean distance to the cosine method.
3.4 Unexpected failures with spectral approximations

As we went through the process of data generation, we found rare cases which completely skewed the results of the spectral clean-up process. In these cases, there is a spectral fitness drop from above 95% to below 50%. LaPlacian, however, continues to perform well even with the “bad data”.

We weren’t able to understand why the bad data was being generated, but we did investigate the possibility of using incorrect eigenvalues. When we retrieved the two largest eigenvalues and eigenvectors from $\tilde{L}$ in order to perform spectral, we were concerned that the MATLAB function returned the two largest positive eigenvalues, not the two largest in magnitude. We did confirm that we did use the two largest magnitude eigenvalues and we ruled this out as a possible problem.

3.5 Accurate data quality analysis

Establishing a consistent way of analyzing the quality of the data was an issue that had not yet been resolved. For example, a priority might be determining the number of rows that had many edges while being located in a region of low probability. This misclassification rate could have an influence on 2-means; with a high misclassification rate, the data is so malformed that algorithms could not even be expected to work properly.
Appendix – MATLAB Scripts

Note: MATLAB Statistics Toolbox is required to run these scripts.

mainScript.m

% Main script
% Runs singleTrial multiple times and compiles results into the result matrix.
% result is an M x 8 matrix: M is the number of times the main for loop
% runs. Each row contains the following information: q1, q2, and five
% average fitnesses (P, Mhat, LHat, Shat, misclassified). See singleTrial
% for more information.

result = [];

n = 1000; % size of matrix
pro = .5; % proportion of q1:q2
start = .01; % starting range of q1
step = .01; % granularity
finish = .95; % ending range of q1

for x = start*n : step*n : finish*n
    % progress
    q1 = x / n;
    q2 = q1 * pro;
    runs = 10; % number of runs averaged
    avg = [0 0 0 0 0];
    for i = 1 : runs
        [fit,badm] = singleTrial(n,q1,q2);
        while(q1 > .1 && fit(1,4) < .6) % re-does run for bad spectral data
            [fit,badm] = singleTrial(n,q1,q2);
        end
        avg = avg + fit;
    end
    avg = avg ./ runs;
    result = [result; q1 q2 avg];
end

result % output result

singleTrial.m

function [fit,badm] = singleTrial(n, q1, q2)
% Runs a single trial for given matrix size, and q1, q2.
% fit is a 1 x 5 matrix containing the fitness of P, Mhat, Lhat, Shat,
% and misclassified.
% P = raw probability matrix
% Mhat = probability matrix injected with degree information
% Lhat = Mhat after Laplacian transformation
% Shat = Lhat after spectral approximation
% misclassified = how bad the raw data is in misclassified rows

fit = [0 0 0 0 0];

% Raw probability matrix
P = ProbMat(n, q1, q2);
fit(1,1) = fitness(kmeans(hattifier(P),2));

% Power-law degree info injected
D = dMaker(P); % make D
M = D'*D;
%M = M / (diag(D)' + diag(D)); % matrix normalization
Mhat = hattifier(M);
fit(1,2) = fitness(kmeans(Mhat,2));
% LaPlacian
Dhalf = D^(-1/2);
Lhat = DHalf*Mhat*DHalf;
fit(1,3) = fitness(kmeans(Lhat, 2));

% Spectral
opts.disp = 0;
[a,b] = eigs(Lhat,2,'LM',opts); % returns two highest magnitude eigenvectors and values
Shat = a * b * a';
fit(1,4) = fitness(kmeans(Shat,2));
badm = [];
if(fit(1,4) < .9)
    badm = Lhat;
end

% Misclassification
fit(1,5) = misclassified(P, q1, q2);

ProbMat.m
function P = ProbMat(n,q1,q2)
p1 = ones(n/2) * q1;
p2 = ones(n/2) * q2;
p4 = p1;
p3 = p2;
P = [p1 p2; p3 p4];

hattifier.m
function A = hattifier(P)
% Rounds input matrix to 0's and 1's
a = rand(size(P, 1)) < P;
A = triu(a) + triu(a,1)';

fitness.m
function f = fitness(v)
n = length(v);
comp1 = ones(n/2,1);
comp2 = ones(n/2,1) + 1;
% compare first half of values to 1's, second half to 2's
f1 = v(1:n/2) == comp1;
f2 = v((n/2)+1:n) == comp2;
% compare first half of values to 2's, second half to 1's
g1 = v(1:n/2) == comp2;
g2 = v((n/2)+1:n) == comp1;
if(sum(f1) > sum(g1)) %if true, ones in first n/2 entries
    f = sum(f1) + sum(f2);
else
    f = sum(g1) + sum(g2);
end
f = f/n;
```matlab
function D = dMaker(P)
% Takes a probability matrix and generates matrix D
% D represents degree information according to powerlaw
% D is a diagonal matrix with di = degree of node i

n = size(P, 1);
c = 0;
for i = 1:n
    c = c + 1/i^3;
end

v = 1:n;
oldv = c./v.^3;
col = 1;
pointer = 0;
d = ones(1,n);
while (floor(oldv(1,col)) > 0)
    d(pointer+1 : pointer+floor(oldv(1,col))) = col;
    pointer = pointer + floor(oldv(1,col));
    col = col + 1;
end
newIndx = randperm(n);
v(1:n) = d(newIndx(1:n));
D = diag(v, 0);
```
Appendix – Table of Results

mainScript.m run at n=1000, runs=10, start=.005, end=.95 with different step sizes.

<table>
<thead>
<tr>
<th>q1</th>
<th>q2</th>
<th>P</th>
<th>M</th>
<th>L</th>
<th>Ls</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.005</td>
<td>0.0025</td>
<td>0.4963</td>
<td>0.5052</td>
<td>0.4919</td>
<td>0.5009</td>
</tr>
<tr>
<td>0.01</td>
<td>0.005</td>
<td>0.5044</td>
<td>0.4988</td>
<td>0.4704</td>
<td>0.5562</td>
</tr>
<tr>
<td>0.015</td>
<td>0.0075</td>
<td>0.4787</td>
<td>0.503</td>
<td>0.585</td>
<td>0.8277</td>
</tr>
<tr>
<td>0.02</td>
<td>0.01</td>
<td>0.5834</td>
<td>0.4973</td>
<td>0.8924</td>
<td>0.8903</td>
</tr>
<tr>
<td>0.025</td>
<td>0.0125</td>
<td>0.8011</td>
<td>0.4935</td>
<td>0.9324</td>
<td>0.5288</td>
</tr>
<tr>
<td>0.03</td>
<td>0.015</td>
<td>0.8983</td>
<td>0.4978</td>
<td>0.956</td>
<td>0.9337</td>
</tr>
<tr>
<td>0.04</td>
<td>0.02</td>
<td>0.9479</td>
<td>0.5013</td>
<td>0.9767</td>
<td>0.9761</td>
</tr>
<tr>
<td>0.05</td>
<td>0.025</td>
<td>0.9707</td>
<td>0.5006</td>
<td>0.9891</td>
<td>0.9883</td>
</tr>
<tr>
<td>0.06</td>
<td>0.03</td>
<td>0.9815</td>
<td>0.5957</td>
<td>0.9948</td>
<td>0.9943</td>
</tr>
<tr>
<td>0.07</td>
<td>0.035</td>
<td>0.9897</td>
<td>0.647</td>
<td>0.9974</td>
<td>0.9969</td>
</tr>
<tr>
<td>0.08</td>
<td>0.04</td>
<td>0.9936</td>
<td>0.6436</td>
<td>0.9989</td>
<td>0.9992</td>
</tr>
<tr>
<td>0.09</td>
<td>0.045</td>
<td>0.9981</td>
<td>0.6636</td>
<td>0.9996</td>
<td>0.9995</td>
</tr>
<tr>
<td>0.1</td>
<td>0.05</td>
<td>0.9977</td>
<td>0.6382</td>
<td>0.9998</td>
<td>0.9997</td>
</tr>
<tr>
<td>0.11</td>
<td>0.055</td>
<td>0.9989</td>
<td>0.5361</td>
<td>0.9998</td>
<td>0.9998</td>
</tr>
<tr>
<td>0.12</td>
<td>0.06</td>
<td>0.9993</td>
<td>0.6032</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0.13</td>
<td>0.065</td>
<td>1</td>
<td>0.6427</td>
<td>0.9999</td>
<td>0.9999</td>
</tr>
<tr>
<td>0.14</td>
<td>0.07</td>
<td>0.9998</td>
<td>0.7508</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0.15</td>
<td>0.075</td>
<td>0.9998</td>
<td>0.6535</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0.16</td>
<td>0.08</td>
<td>1</td>
<td>0.8013</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0.17</td>
<td>0.085</td>
<td>1</td>
<td>0.6012</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0.18</td>
<td>0.09</td>
<td>1</td>
<td>0.697</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0.19</td>
<td>0.095</td>
<td>1</td>
<td>0.6016</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0.2</td>
<td>0.1</td>
<td>1</td>
<td>0.8007</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0.21</td>
<td>0.105</td>
<td>1</td>
<td>0.798</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0.22</td>
<td>0.11</td>
<td>1</td>
<td>0.6975</td>
<td>0.9997</td>
<td>0.9997</td>
</tr>
<tr>
<td>0.23</td>
<td>0.115</td>
<td>1</td>
<td>0.7047</td>
<td>0.9996</td>
<td>0.9994</td>
</tr>
<tr>
<td>0.24</td>
<td>0.12</td>
<td>1</td>
<td>0.7486</td>
<td>0.9995</td>
<td>0.9994</td>
</tr>
<tr>
<td>0.25</td>
<td>0.125</td>
<td>1</td>
<td>0.6965</td>
<td>0.9993</td>
<td>0.9993</td>
</tr>
<tr>
<td>0.26</td>
<td>0.13</td>
<td>1</td>
<td>0.7485</td>
<td>0.9991</td>
<td>0.9991</td>
</tr>
<tr>
<td>0.27</td>
<td>0.135</td>
<td>1</td>
<td>0.6951</td>
<td>0.999</td>
<td>0.999</td>
</tr>
<tr>
<td>0.28</td>
<td>0.14</td>
<td>1</td>
<td>0.7498</td>
<td>0.9991</td>
<td>0.9989</td>
</tr>
<tr>
<td>0.29</td>
<td>0.145</td>
<td>1</td>
<td>0.6999</td>
<td>0.998</td>
<td>0.998</td>
</tr>
<tr>
<td>0.3</td>
<td>0.15</td>
<td>1</td>
<td>0.6546</td>
<td>0.9977</td>
<td>0.9979</td>
</tr>
<tr>
<td>0.31</td>
<td>0.155</td>
<td>1</td>
<td>0.7457</td>
<td>0.9982</td>
<td>0.9982</td>
</tr>
<tr>
<td>0.32</td>
<td>0.16</td>
<td>1</td>
<td>0.7965</td>
<td>0.9978</td>
<td>0.9978</td>
</tr>
<tr>
<td>0.33</td>
<td>0.165</td>
<td>1</td>
<td>0.7515</td>
<td>0.9974</td>
<td>0.9974</td>
</tr>
<tr>
<td>0.34</td>
<td>0.17</td>
<td>1</td>
<td>0.7947</td>
<td>0.9969</td>
<td>0.9964</td>
</tr>
<tr>
<td>0.35</td>
<td>0.175</td>
<td>1</td>
<td>0.6029</td>
<td>0.9965</td>
<td>0.9966</td>
</tr>
<tr>
<td>0.36</td>
<td>0.18</td>
<td>1</td>
<td>0.5931</td>
<td>0.996</td>
<td>0.9956</td>
</tr>
<tr>
<td>0.37</td>
<td>0.185</td>
<td>1</td>
<td>0.7992</td>
<td>0.9969</td>
<td>0.9963</td>
</tr>
<tr>
<td>0.38</td>
<td>0.19</td>
<td>1</td>
<td>0.7444</td>
<td>0.9963</td>
<td>0.9969</td>
</tr>
<tr>
<td>0.39</td>
<td>0.195</td>
<td>1</td>
<td>0.7493</td>
<td>0.9962</td>
<td>0.9958</td>
</tr>
<tr>
<td>0.4</td>
<td>0.2</td>
<td>1</td>
<td>0.6489</td>
<td>0.9933</td>
<td>0.994</td>
</tr>
<tr>
<td>0.41</td>
<td>0.205</td>
<td>1</td>
<td>0.8455</td>
<td>0.9939</td>
<td>0.9939</td>
</tr>
<tr>
<td>0.42</td>
<td>0.21</td>
<td>1</td>
<td>0.8446</td>
<td>0.994</td>
<td>0.9937</td>
</tr>
<tr>
<td>0.43</td>
<td>0.215</td>
<td>1</td>
<td>0.7944</td>
<td>0.9927</td>
<td>0.9934</td>
</tr>
<tr>
<td>0.44</td>
<td>0.22</td>
<td>1</td>
<td>0.7481</td>
<td>0.9944</td>
<td>0.9938</td>
</tr>
<tr>
<td>0.45</td>
<td>0.225</td>
<td>1</td>
<td>0.8918</td>
<td>0.9943</td>
<td>0.9942</td>
</tr>
<tr>
<td>0.46</td>
<td>0.23</td>
<td>1</td>
<td>0.7443</td>
<td>0.9942</td>
<td>0.9941</td>
</tr>
<tr>
<td>0.47</td>
<td>0.235</td>
<td>1</td>
<td>0.791</td>
<td>0.9927</td>
<td>0.9928</td>
</tr>
<tr>
<td>0.48</td>
<td>0.24</td>
<td>1</td>
<td>0.8869</td>
<td>0.9941</td>
<td>0.9936</td>
</tr>
<tr>
<td>0.49</td>
<td>0.245</td>
<td>1</td>
<td>0.792</td>
<td>0.9922</td>
<td>0.992</td>
</tr>
<tr>
<td>0.5</td>
<td>0.25</td>
<td>1</td>
<td>0.8888</td>
<td>0.9875</td>
<td>0.9871</td>
</tr>
<tr>
<td>0.55</td>
<td>0.275</td>
<td>1</td>
<td>0.8886</td>
<td>0.9879</td>
<td>0.987</td>
</tr>
<tr>
<td>0.6</td>
<td>0.3</td>
<td>1</td>
<td>0.8805</td>
<td>0.9868</td>
<td>0.9879</td>
</tr>
<tr>
<td>0.65</td>
<td>0.325</td>
<td>1</td>
<td>0.9245</td>
<td>0.9845</td>
<td>0.9839</td>
</tr>
<tr>
<td>0.7</td>
<td>0.35</td>
<td>1</td>
<td>0.9258</td>
<td>0.973</td>
<td>0.9719</td>
</tr>
<tr>
<td>0.75</td>
<td>0.375</td>
<td>1</td>
<td>0.9731</td>
<td>0.9725</td>
<td>0.9728</td>
</tr>
<tr>
<td>0.8</td>
<td>0.4</td>
<td>1</td>
<td>0.9721</td>
<td>0.9735</td>
<td>0.9725</td>
</tr>
<tr>
<td>0.85</td>
<td>0.425</td>
<td>1</td>
<td>0.971</td>
<td>0.9725</td>
<td>0.9715</td>
</tr>
<tr>
<td>0.9</td>
<td>0.45</td>
<td>1</td>
<td>0.9396</td>
<td>0.9733</td>
<td>0.9722</td>
</tr>
<tr>
<td>0.95</td>
<td>0.475</td>
<td>1</td>
<td>0.9205</td>
<td>0.9215</td>
<td>0.9245</td>
</tr>
</tbody>
</table>
References

1. Meetings with John Hopcroft and Bart Selman, professors in the Computer Science Department at Cornell University.


