Exploration of Sea Ice Concentrations using Graph Metrics

Ryan Dougherty
Computer Science
Arizona State University
Tempe, AZ 85253, USA

March 30, 2015

*Funded by the Fulton Undergraduate Research Initiative (FURI), Ira A. Fulton Schools of Engineering, Arizona State University.
## Contents

1. **Introduction** 5  
   - 1.1 Data Gathering 5  
   - 1.2 Relations with Graphs 5  
   - 1.3 Motivation 6  
   - 1.4 Past Work 6  
   - 1.5 Thesis Outline 6  
   - 1.6 Module Documentation 7  

2. **Closeness Centrality and Degrees** 8  
   - 2.1 Closeness Centrality 8  
   - 2.2 Closeness Centrality Results 10  
   - 2.3 Vertex Degrees 11  

3. **Vertex Intersections** 14  
   - 3.1 Vertex Intersection Results 14  

4. **Betweenness Centrality** 21  
   - 4.1 Betweenness Centrality 21  
   - 4.2 Betweenness Centrality Results 22  

5. **Induced Time Lag** 27  
   - 5.1 Vertex Similarity 27  
   - 5.2 Cosine Similarity 28  
   - 5.3 Cosine Similarity Results 29  

6. **Characteristic Path Length and Connected Components** 33  
   - 6.1 CPL Results 36  

7. **Computational Advancements** 39  
   - 7.1 Graph Density 39  
   - 7.2 Characteristic Path Length Running Times 39  
   - 7.3 Parallelized Betweenness Centrality 39  

8. **Future Work** 43  

9. **Conclusion** 46  

10. **Acknowledgments** 47  

Appendices 48  

A. **Definitions** 48  

B. **Code Module Documentation** 50  

References 52
LIST OF TABLES

List of Figures

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Connected Component Sizes, 1979</td>
</tr>
<tr>
<td>2</td>
<td>Maximum Closeness Vertex X and Y coordinate per year</td>
</tr>
<tr>
<td>3</td>
<td>Maximum Closeness with GSFC overlay</td>
</tr>
<tr>
<td>4</td>
<td>Power Law Distribution Exponent Values</td>
</tr>
<tr>
<td>5</td>
<td>Power Law Distribution $R^2$ Values</td>
</tr>
<tr>
<td>6</td>
<td>Power Law Degree Distribution, 1988</td>
</tr>
<tr>
<td>7</td>
<td>$NGIN(i)$ for $2 \leq i \leq 27$</td>
</tr>
<tr>
<td>8</td>
<td>4 Vertex Intersections with Map Overlay</td>
</tr>
<tr>
<td>9</td>
<td>10 Vertex Intersections with Map Overlay</td>
</tr>
<tr>
<td>10</td>
<td>25 Vertex Intersections with Map Overlay</td>
</tr>
<tr>
<td>11</td>
<td>27 Vertex Intersections with Map Overlay</td>
</tr>
<tr>
<td>12</td>
<td>Betweenness Centrality values, 1983</td>
</tr>
<tr>
<td>13</td>
<td>Betweenness Centrality values, 1983</td>
</tr>
<tr>
<td>14</td>
<td>Betweenness Centrality values, Max Set, 1983</td>
</tr>
<tr>
<td>15</td>
<td>Max Betweenness values per year</td>
</tr>
<tr>
<td>16</td>
<td>Maximum Cosine Similarity, Yearly Comparison</td>
</tr>
<tr>
<td>17</td>
<td>Average Cosine Similarity, Yearly Comparison</td>
</tr>
<tr>
<td>18</td>
<td>CPL Results Per Year</td>
</tr>
<tr>
<td>19</td>
<td>Maximum CPL Per Year</td>
</tr>
<tr>
<td>20</td>
<td>CPL Percentages Per Year</td>
</tr>
<tr>
<td>21</td>
<td>Number of Edges</td>
</tr>
<tr>
<td>22</td>
<td>CPL Running Times, Old vs. New Algorithm</td>
</tr>
</tbody>
</table>

List of Tables

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Pair-wise Vertex Intersections, [1979, 2005] × [1979, 1991]</td>
</tr>
<tr>
<td>3</td>
<td>Global Intersection Number and Number of Vertices</td>
</tr>
<tr>
<td>4</td>
<td>Max Betweenness values per year</td>
</tr>
<tr>
<td>5</td>
<td>4-Week Maximum Cosine Similarity Values</td>
</tr>
<tr>
<td>6</td>
<td>4-Week Average Cosine Similarity Values</td>
</tr>
<tr>
<td>7</td>
<td>Number of Vertices and Edges per Year</td>
</tr>
<tr>
<td>8</td>
<td>CPL Running Times, Old vs. New Algorithm</td>
</tr>
</tbody>
</table>
Abstract

As an example of “big data,” we consider a repository of Arctic sea ice concentration data collected from satellites over the years 1979-2005. The data is represented by a graph, where vertices correspond to measurement points, and an edge is inserted between two vertices if the Pearson correlation coefficient between them exceeds a threshold. We investigate new questions about the structure of the graph related to betweenness, closeness centrality, vertex degrees, and characteristic path length. We also investigate whether an offset of weeks and years in graph generation results in a cosine similarity value that differs significantly from expected values. Finally, we relate the computational results to trends in Arctic ice.
1 Introduction

The topic relates to working with large sets of data. The data set comes from the National Snow and Ice Data Center (NSIDC), and consists of sea ice concentrations of the Arctic Ocean from 1979 through 2005. In the next two sections, we explain how the data was gathered, and then what connections can be made using them.

1.1 Data Gathering

The data were measured over these years with a Scanning Multichannel Microwave Radiometer (SMMR), and later with technology involving a Special Sensor Microwave/Imager (SMM/I). These sensors divide the Arctic Ocean into a grid of 448 × 304 regions, each one representing a 25km × 25km area. Then the concentration of the ice (i.e., how much of the surface area was covered by ice) was measured in each region, daily. Any land or missing data were indicated by special values. The daily data were aggregated into weekly deviations from the average value (over the entire year) for each region.

The data set is so large (several hundred megabytes in size) that some calculations require very many computations. Therefore, it is worthwhile to investigate and implement efficient algorithms for processing the data and drawing conclusions from them.

1.2 Relations with Graphs

The data set can be represented as a graph. For a general reference on graph theory topics, see [10]. The 66129 sea ice vertices (those that are not missing nor land data) correspond to the 25km × 25km areas. The edges are created between these vertices as follows: for each pair of vertices \( u \) and \( v \) in the graph with \( u \neq v \), we calculate the Pearson product-moment correlation coefficient [23] of \( u \) and \( v \) using their corresponding values through all weeks of that year. This calculation of \( r \) is given by the following formula:

\[
 r = \frac{\sum_{i=1}^{n}(X_i - \overline{X})(Y_i - \overline{Y})}{\sqrt{\sum_{i=1}^{n}(X_i - \overline{X})^2 \sqrt{\sum_{i=1}^{n}(Y_i - \overline{Y})^2}}}
\]

where \( X_i \) and \( Y_i \) correspond to the value of \( u \) and \( v \) at week \( i \), respectively. \( \overline{X} \) and \( \overline{Y} \) correspond to the average values of \( u \) and \( v \) over the entire year. For calculating over an entire year, \( n = 52 \).

Given as input is a threshold \( r_{\text{threshold}} \). If \( r > r_{\text{threshold}} \), we insert an undirected edge between \( u \) and \( v \). For this thesis, we select \( r_{\text{threshold}} = 0.9 \). After using this value as the threshold, the resulting graphs that are studied have between 1 and 5 million edges each.

Using this graph representation allows us to examine and identify unique properties that the graph exhibits. Some of these include the characteristic path length (CPL), connected components, clustering coefficients, centrality measurements, and more. All of these allow one to draw conclusions about the Arctic sea ice based on the trends that they exhibit.

Also, there is very fine-grained control over what graphs are created. For example, one can choose graphs to be built for weeks 8-12 of years 1985-1988. All of the graphs mentioned in this thesis, unless otherwise noted, consist of weeks 1-52 (the entire year) over each year between 1979 and 2005, resulting in 27 graphs. Changes in the data set over time may become more prominent.
in this way. For example, in Section 6, we show that the characteristic path length (CPL) of the graphs decreases over time, whereas one graph would only give one value of the CPL. With multiple graphs, not only can we analyze the numerical values of the results and what implications they have, but also see how these values change over time.

1.3 Motivation

The compelling question relating to this project is “What significant information can we gather from graph representations of the data?” The topic is interesting because not only is it an insight into real-world data analysis and academic work, but also one can understand approaches to global problems others have investigated, and what contributions can be made. The main goals of the thesis and the study are to explore topics in applications of graph theory, handling big data sets, and constructing and developing data structures to analyze the data set of Arctic sea ice concentrations.

1.4 Past Work

Patterson’s undergraduate honors thesis [16] laid the fundamental groundwork for implementation and analysis of the data set. Much of her work centered around creating various visual representations of the graph, and she did this by plotting degree distributions, connected components by colors, and more. The Eclipse IDE and the A2C2 Advanced Computing Cluster [3], hosted at ASU, for doing the large multithreaded computations were involved in building the graphs. Some of the other aspects of the data set investigated were:

- Percent surface covered by ice over time for a particular connected component.
- Similarities between properties found in the sea ice data and those of small-world graphs/networks, whose hypotheses were tested using statistical analysis.
- Creation of modules for constructing “random graphs” (specifically, Erdős-Rényi [12]). She then compared, across various aggregations of vertices by physical location, the characteristic path length, global clustering coefficient, and more.

However, there are properties of the graphs that are even more useful for climatologists. For example, the work in [16] only dealt with aggregations of $2 \times 2$, $4 \times 4$, and $8 \times 8$ vertices, purely for speeding up the creation of graphs. Since the data sets are already aggregated by week, analyzing the graphs without aggregation provides the closest possible representation of the actual data set. Also, many of the graphs used in [16] were constructed over all 27 years provided, and a more “fine-grained” analysis is desired.

1.5 Thesis Outline

The rest of this thesis is organized as follows.

In Section 2, we investigate connected components and closeness centrality, and define an algorithm that relates these two metrics with physical areas of ice as studied in [16]. Also, we investigate degrees of the vertices in the graphs, and make an association with a subset of small-world graphs.
In Section 3, we look at vertex intersections between the largest connected components of a selection of graphs; specifically, how many vertices are within this component of each graph in the selection. We consider pair-wise intersections as well as a global intersections.

In Section 4, we look at betweenness centrality, and show how the NSIDC graphs’ distributions of betweenness values follows that commonly found in practice for small-world graphs. We then create images that give a visual representation of the betweenness values.

In Section 5, we investigate two measures of graph similarity and reason about their tractability. We then analyze one of these metrics, and relate a well known model of graphs to the NSIDC sea ice graphs through the metric. In the results, we first compare the yearly graphs, and then create new “weekly” graphs that have an offset of a number of weeks. We then relate the conclusions drawn from these computed values back to the NSIDC sea ice data.

In Section 6, we give improvements (in terms of memory usage and computation time) to well-known algorithms for computing the characteristic path length of a graph based on the constructions of the yearly graphs. We then calculate upper and lower bounds of characteristic path length values, and derive conclusions based on the actual values and the computed upper bound values.

In Section 7, we propose improvements to the betweenness centrality algorithm as given in Section 4. We then give results on the improved algorithms that are described in Sections 4 and 6 in terms of running time.

In Section 8, we give a list of items that can enhance one’s understanding of the sea ice data set, through investigating new techniques in designing algorithms for performance benefits and thinking about a new method of analyzing the data set.

In Section 9, we summarize the results presented in this thesis, and what connections to the data set have been made.

1.6 Module Documentation

Code modules that are involved in the computations in this thesis have documentation in Appendix B. All of the modules are built for the Arctic data set.
2 Closeness Centrality and Degrees

We discuss measures used for analyzing trends in large networks and graphs: closeness centrality, and vertex degrees and their relevance to power-law distributions. The importance of studying these measures is to characterize the NSIDC graphs in various ways.

A connected component \( CC = (V_{CC}, E_{CC}) \) is defined to be a subgraph of \( G = (V_G, E_G) \) such that:

1. \( V_{CC} \subseteq V_G \) and \( E_{CC} \subseteq E_G \)
2. \( \forall v \in V_{CC}, \exists v' \in V_{CC} \setminus \{v\} \) with some \( e \in E_{CC} \) connecting \( v \) and \( v' \).

Informally, a connected component is a set of vertices in the graph where all of the vertices in that set are connected to some other vertex also in the set. In Figure 1, we plot the largest 50 connected components by size in sorted order of the 1979 graph; the vertical axis is on a logarithmic scale with the number of vertices in that component. All of the other graphs have very similar structure with respect to their components. For example, the same year has a total of 727 components, the large majority of which have 2 vertices.

A graph can be undirected or directed; the former has edges where a path can traverse both directions through the edge, and the latter has edges where a path can only traverse in the direction of the edge. The construction for the NSIDC graphs has the edges be undirected. We can easily see that all of the connected components are “islands” (i.e., where any pair of connected components do not have vertices in common).

2.1 Closeness Centrality

Closeness centrality is a measure of how “close” a particular vertex is to all other vertices in the same component in terms of how many edges it is from all other reachable vertices. Formally, closeness is defined for a vertex \( v \in V_i \), where \( V_i \) induces a connected component of an undirected graph \( G \) as:

\[
\text{Closeness}(v) = \frac{1}{\sum_{j \neq v, j \in V_i} d(v, j)}
\]

where \( d(v, j) \) is the length of the shortest path between \( v \) and \( j \) (some authors multiply \( \text{Closeness}(v) \) by \( N - 1 \), where \( N \) is the number of vertices in the connected component. This is just a normalization factor, since \( N - 1 \) is constant). The vertex with the highest closeness value is the most centrally located vertex, in terms of the graph:

\[
v_{\text{centermost}} = \max_{v \in V_i} \left\{ \text{Closeness}(v) \right\} = \max_{v \in V_i} \left\{ \frac{1}{\sum_{j \neq v, j \in V_i} d(v, j)} \right\}.
\]

If we consider components of size 2, each vertex in that small connected component would have a closeness value of 1 (the maximum possible value), which does not lead to any useful conclusions. Therefore, we only consider the largest connected component. Algorithm 1 evaluates closeness for all \( v \in V \) of a graph \( G \), as detailed in \cite{17}. 
Input: $G = (V, E)$ undirected, unweighted graph

Result: $CC(v) = \text{exact closeness centrality value for } v \in V$.

for $s \in V$ do
  Q ← empty queue;
  $d[v] \leftarrow \infty, \forall v \in V \setminus \{s\}$;
  push $s \rightarrow Q$;
  $d[s] \leftarrow 0$;
  $far[s] \leftarrow 0$;
  while $Q$ not empty do
    pop $v \leftarrow Q$;
    forall the $w \in \Gamma_G\{v\}$ do
      if $d[w] = \infty$ then
        push $w \rightarrow Q$;
        $d[w] \leftarrow d[v] + 1$;
        $far[s] \leftarrow far[s] + d[w]$;
      end
    end
  end
  $CC[s] = \frac{1}{far[s]}$;
end
return $CC$;

Algorithm 1: Exact Closeness Centrality
2.2 Closeness Centrality Results

In Figure 2, we present the year, x coordinate, and y coordinate of the vertex of the grid with highest closeness value in that year.

As we can see, there is not much of a correlation over time in terms of the x and y coordinates of this center vertex alone. However, for the large majority of the years, the x coordinate is larger (and sometimes much larger) than the y coordinate. Disregarding the years where the x and y coordinates are similar in value, the values of the x and y coordinates by themselves are nearly the same: the x coordinate is largely between 300 and 375, and the y coordinate is largely between 100 and 200. These results imply that the centermost vertex, for most years, is within the corresponding physical bounds. However, by looking at all the centermost vertices in aggregate, we can see that these vertices are physically close together, showing that the centermost vertex does not change physical location drastically.

In Figure 3, we plot the results from Figure 2 onto an image (as white coordinates). We then apply a map overlay (in dark blue) provided by the Goddard Space Flight Center (GSFC)\[1\] of the land masses surrounding the Arctic Ocean that are in the NSIDC data set. The motivation for doing this is to see what implications the values from the algorithms have on the physical land masses. We can see that the large majority of the points are in the regions near the southwest and northeast corners of Greenland; specifically, the southwest corner corresponds to Baffin Bay and Labrador Sea. These vertices are considered to be the centermost vertices of the graph because they have the highest closeness value of the entire graph. What is interesting in the southwest corner case is that all of the points are physically close to each other. The years to which these points correspond are (in order): 1981, 1986, 1988, 1990, 1992, 1993, 1994, 1996, 1998, 1999, 2002, and 2004.

In [16], a connected component was identified with this particular region along with measurements over time in terms of the percent surface area covered by sea ice. In that, there was two instances...
of a rise and fall in concentration: during 1982-1985, and 1988-1995. However, there were no measurements of percent sea ice coverage outside of these year ranges.

2.3 Vertex Degrees

We now focus on the degrees of vertices in the graphs. The work in [16] concluded that the graphs of the sea ice data set resemble small-world graphs. We extend that analysis by showing that the NSIDC graphs can be classified more specifically as a subclass of small-world graphs - scale-free. Scale-free networks arise in many fields - one example is the study of the topology of web pages [20]. This type of network is characterized for a graph $G$ as when its the degrees for each of its vertices follow a power-law distribution, which is of the form:

$$P(k) \propto k^{-\gamma}$$

where $P(k)$ corresponds to the probability of the vertices in $G$ having $k$ edges to other vertices, and $\gamma$ is the constant in which the distribution fits best. This $\gamma$ is what is of interest in scale-free networks. For example, most scale-free networks have $2 < \gamma < 3$, but sometimes $\gamma$ may be outside of this range [14]. The characterization of $G$ as a scale-free network relies on the regression constant (how well the degree distributions match a power law).

In Figures 4 and 5, we plot the power law exponents and $R^2$ values. Since $R^2 > 0.7$ for all years, we can conclude that the graphs are in fact scale-free. What is interesting is the exponent values over time in Figure 4 matched with the $R^2$ values in Figure 5 for some of the years. For example,
the years 1982-1984 and 2003-2005 both had the exponent and $R^2$ values decrease. Also, the shape of the plots is similar for the years 1985-2002.

We now give an example of one of the power law distributions; namely for the year 1988. In Figure 6, we plot the degree distribution for the graph of that year. On the horizontal axis, we plot the degree of the vertex, and on the horizontal axis we plot the frequency of that particular degree. We also plot the best-fit (power law) line that corresponds to that of Figure 4. It does not appear as though this line correlates well with the distribution for a degree of $\leq 50$; however, one can see that Figure 5 shows that there is a very high correlation among all of the graphs.
Closeness Centrality and Degrees

Figure 5: Power Law Distribution $R^2$ Values

Figure 6: Power Law Degree Distribution, 1988
3 Vertex Intersections

What is also important to study about the vertices is whether they remain in the largest connected component of the graph over each year; and if not all years, how many years (and which ones) that vertex does. If a vertex \( v \) remains in this connected component across all years, then it is structurally integral to the graph’s largest connected component.

We analyze vertex intersections in two different ways: for all pair-wise intersections, and a global intersection. For pair-wise, we take every pair of graphs and compute the largest connected component, as well as how many vertices are within both of them.

For global intersections, we take all 27 graphs, and repeated the same process as for pair-wise, but compute one value, which gives the number of vertices that were in the largest connected component for all years.

3.1 Vertex Intersection Results

We created code module `VertexIntersection.java` for handling the pair-wise and global computations. In Tables 1 and 2, we show the number of vertices that are within the largest connected component of both of the compared graphs (if the graphs are the same, then there is no value given). As one can see, the number for each pair is very large; most values in both tables are over 30000. From these results, we can conclude that the difference in terms of this connected component between any pair of graphs does not alter much in terms of which vertices are included.

Now we investigate global intersections. To do so, we looked at each of the years’ largest connected component, and counted, for each vertex, exactly how many years that vertex appeared. We outputted files of the form `outputIntersection<i>.txt` for \( 2 \leq i \leq 27 \) which lists the coordinates of the vertices that are involved in exactly \( i \) intersections.

We define “Global Intersection Number” for a particular vertex \( v \), \( GIN(v) \), to be the exact number of graphs for which it is in the largest connected component. Clearly, \( 0 \leq GIN(v) \leq 27 \) for any vertex \( v \), but since we are interested in the largest connected component, then we have \( 1 \leq GIN(v) \). Also, since we are considering any number of intersections across graphs, we have \( 2 \leq GIN(v) \). In Table 3 and Figure 7 we look at the number of vertices that have a value of \( GIN(v) = i \) for \( 2 \leq i \leq 27 \) - call this value \( NGIN(i) \). It is interesting to note that the values decrease and then increase over time (with the turnaround point at \( i = 13, 15 \)).

However, what is more interesting is the last value: \( NGIN(27) = 18511 \). The number of vertices that are present within the largest connected component over all graphs is over half of all of the vertices that are connected to at least one other vertex. In terms of the sea ice, the corresponding physical locations of the vertices that are in this set of 18511 vertices has that they are always central to the entire sea ice system.

We now generate images to visually show what is happening with these global intersections. For \( 2 \leq i \leq 27 \), we generated a file called `<i>VertexIntersections.png`. In each image, the points that have a pixel of white correspond to those vertices that are involved in exactly \( i \) intersections, and the dark-green points are the map overlay provided by GSFC. In Figures 8 to 11 we show 4 images: for \( i \in \{4, 10, 25, 27\} \). The images were chosen to show the trend in Table 3 and Figure 7 especially for \( i = 27 \).

We analyze Figure 11 - one can clearly see that the large majority of the points in the Arctic
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1979</td>
<td>30545</td>
<td>30420</td>
<td>30839</td>
<td>30553</td>
<td>30247</td>
<td>31545</td>
<td>30847</td>
<td>31324</td>
<td>31809</td>
<td>32071</td>
<td>32069</td>
<td>31994</td>
<td></td>
</tr>
<tr>
<td>1980</td>
<td>30545</td>
<td>30582</td>
<td>30359</td>
<td>29334</td>
<td>30376</td>
<td>29535</td>
<td>30177</td>
<td>30648</td>
<td>30785</td>
<td>30813</td>
<td>30728</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1981</td>
<td>30420</td>
<td>29992</td>
<td>30417</td>
<td>30446</td>
<td>29354</td>
<td>30343</td>
<td>29694</td>
<td>30239</td>
<td>30665</td>
<td>30804</td>
<td>30909</td>
<td>30766</td>
<td></td>
</tr>
<tr>
<td>1982</td>
<td>30839</td>
<td>30582</td>
<td>30417</td>
<td>30733</td>
<td>29831</td>
<td>30971</td>
<td>30007</td>
<td>30826</td>
<td>31143</td>
<td>31395</td>
<td>31352</td>
<td>31457</td>
<td></td>
</tr>
<tr>
<td>1983</td>
<td>30553</td>
<td>30359</td>
<td>30446</td>
<td>30733</td>
<td>29912</td>
<td>30842</td>
<td>29965</td>
<td>30625</td>
<td>31119</td>
<td>31228</td>
<td>31161</td>
<td>31176</td>
<td></td>
</tr>
<tr>
<td>1984</td>
<td>30247</td>
<td>29334</td>
<td>29354</td>
<td>29831</td>
<td>29912</td>
<td>31011</td>
<td>29533</td>
<td>29971</td>
<td>30063</td>
<td>31628</td>
<td>31682</td>
<td>30620</td>
<td></td>
</tr>
<tr>
<td>1985</td>
<td>31454</td>
<td>30763</td>
<td>30343</td>
<td>30971</td>
<td>30842</td>
<td>30111</td>
<td>30510</td>
<td>31237</td>
<td>31491</td>
<td>32135</td>
<td>31687</td>
<td>31880</td>
<td></td>
</tr>
<tr>
<td>1986</td>
<td>30847</td>
<td>29535</td>
<td>29694</td>
<td>30007</td>
<td>29965</td>
<td>29533</td>
<td>30510</td>
<td>30632</td>
<td>30932</td>
<td>30936</td>
<td>30962</td>
<td>30871</td>
<td></td>
</tr>
<tr>
<td>1987</td>
<td>31324</td>
<td>30177</td>
<td>30239</td>
<td>30826</td>
<td>30625</td>
<td>29971</td>
<td>31237</td>
<td>30632</td>
<td>31631</td>
<td>31618</td>
<td>31757</td>
<td>31589</td>
<td></td>
</tr>
<tr>
<td>1988</td>
<td>31809</td>
<td>30648</td>
<td>30665</td>
<td>31143</td>
<td>31119</td>
<td>30747</td>
<td>31491</td>
<td>30932</td>
<td>31631</td>
<td>32344</td>
<td>32546</td>
<td>32487</td>
<td></td>
</tr>
<tr>
<td>1989</td>
<td>32071</td>
<td>30785</td>
<td>30804</td>
<td>31395</td>
<td>31228</td>
<td>30553</td>
<td>32135</td>
<td>30936</td>
<td>31618</td>
<td>32344</td>
<td>32928</td>
<td>32796</td>
<td></td>
</tr>
<tr>
<td>1990</td>
<td>32069</td>
<td>30813</td>
<td>30909</td>
<td>31352</td>
<td>31161</td>
<td>30774</td>
<td>31687</td>
<td>30962</td>
<td>31757</td>
<td>32546</td>
<td>32928</td>
<td>32796</td>
<td></td>
</tr>
<tr>
<td>1991</td>
<td>31994</td>
<td>30728</td>
<td>30766</td>
<td>31457</td>
<td>31176</td>
<td>30620</td>
<td>31880</td>
<td>30871</td>
<td>31589</td>
<td>32487</td>
<td>32796</td>
<td>32767</td>
<td></td>
</tr>
<tr>
<td>1992</td>
<td>32127</td>
<td>30818</td>
<td>31006</td>
<td>31533</td>
<td>31469</td>
<td>30603</td>
<td>32188</td>
<td>31149</td>
<td>31831</td>
<td>32638</td>
<td>33213</td>
<td>33015</td>
<td></td>
</tr>
<tr>
<td>1993</td>
<td>32155</td>
<td>30904</td>
<td>30997</td>
<td>31538</td>
<td>31294</td>
<td>30609</td>
<td>32136</td>
<td>31059</td>
<td>31776</td>
<td>32608</td>
<td>33105</td>
<td>33109</td>
<td></td>
</tr>
<tr>
<td>1994</td>
<td>32041</td>
<td>30821</td>
<td>30747</td>
<td>31338</td>
<td>31186</td>
<td>30406</td>
<td>31977</td>
<td>30705</td>
<td>31455</td>
<td>32338</td>
<td>32872</td>
<td>32746</td>
<td></td>
</tr>
<tr>
<td>1995</td>
<td>32338</td>
<td>30960</td>
<td>30853</td>
<td>31554</td>
<td>31286</td>
<td>30676</td>
<td>32140</td>
<td>30995</td>
<td>31789</td>
<td>32712</td>
<td>33247</td>
<td>33167</td>
<td></td>
</tr>
<tr>
<td>1996</td>
<td>31742</td>
<td>30539</td>
<td>30450</td>
<td>31012</td>
<td>30812</td>
<td>30357</td>
<td>31647</td>
<td>30550</td>
<td>31219</td>
<td>32032</td>
<td>32431</td>
<td>32508</td>
<td></td>
</tr>
<tr>
<td>1997</td>
<td>32119</td>
<td>30710</td>
<td>30842</td>
<td>30419</td>
<td>31215</td>
<td>30472</td>
<td>31975</td>
<td>30944</td>
<td>31752</td>
<td>32407</td>
<td>33063</td>
<td>32839</td>
<td></td>
</tr>
<tr>
<td>1998</td>
<td>32334</td>
<td>30959</td>
<td>31243</td>
<td>31783</td>
<td>31591</td>
<td>30717</td>
<td>32041</td>
<td>31182</td>
<td>32097</td>
<td>32635</td>
<td>33174</td>
<td>33187</td>
<td></td>
</tr>
<tr>
<td>1999</td>
<td>31789</td>
<td>30518</td>
<td>30552</td>
<td>31077</td>
<td>30184</td>
<td>30355</td>
<td>31445</td>
<td>30654</td>
<td>31453</td>
<td>32325</td>
<td>32579</td>
<td>32666</td>
<td></td>
</tr>
<tr>
<td>2000</td>
<td>32500</td>
<td>31275</td>
<td>31182</td>
<td>31827</td>
<td>31588</td>
<td>30907</td>
<td>32717</td>
<td>31344</td>
<td>31997</td>
<td>32845</td>
<td>33577</td>
<td>33140</td>
<td></td>
</tr>
<tr>
<td>2001</td>
<td>32041</td>
<td>30689</td>
<td>30864</td>
<td>31349</td>
<td>31216</td>
<td>30736</td>
<td>31847</td>
<td>31015</td>
<td>31873</td>
<td>32647</td>
<td>32977</td>
<td>33107</td>
<td></td>
</tr>
<tr>
<td>2002</td>
<td>32163</td>
<td>30790</td>
<td>30915</td>
<td>31457</td>
<td>31246</td>
<td>30713</td>
<td>31718</td>
<td>31132</td>
<td>31944</td>
<td>32752</td>
<td>32882</td>
<td>33178</td>
<td></td>
</tr>
<tr>
<td>2003</td>
<td>31977</td>
<td>30517</td>
<td>30457</td>
<td>31203</td>
<td>30964</td>
<td>30238</td>
<td>31578</td>
<td>30825</td>
<td>31655</td>
<td>32314</td>
<td>32754</td>
<td>32890</td>
<td></td>
</tr>
<tr>
<td>2004</td>
<td>31883</td>
<td>30393</td>
<td>30518</td>
<td>31077</td>
<td>30717</td>
<td>30294</td>
<td>31257</td>
<td>30727</td>
<td>31544</td>
<td>32907</td>
<td>32391</td>
<td>32745</td>
<td></td>
</tr>
<tr>
<td>2005</td>
<td>31490</td>
<td>30362</td>
<td>30404</td>
<td>30828</td>
<td>30865</td>
<td>30085</td>
<td>31575</td>
<td>30331</td>
<td>31193</td>
<td>31921</td>
<td>32650</td>
<td>32492</td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Pair-wise Vertex Intersections, [1979, 2005] × [1979, 1991]
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>32127</td>
<td>32155</td>
<td>32041</td>
<td>32338</td>
<td>31742</td>
<td>32119</td>
<td>32334</td>
<td>31789</td>
<td>32500</td>
<td>32041</td>
<td>32163</td>
<td>31977</td>
<td>31883</td>
<td>31490</td>
<td></td>
</tr>
<tr>
<td>30818</td>
<td>30904</td>
<td>30821</td>
<td>30960</td>
<td>30539</td>
<td>30710</td>
<td>30959</td>
<td>30518</td>
<td>31275</td>
<td>30689</td>
<td>30790</td>
<td>30517</td>
<td>30393</td>
<td>30362</td>
<td></td>
</tr>
<tr>
<td>31006</td>
<td>30997</td>
<td>30747</td>
<td>30853</td>
<td>30450</td>
<td>30842</td>
<td>31243</td>
<td>30552</td>
<td>31182</td>
<td>30864</td>
<td>30915</td>
<td>30457</td>
<td>30518</td>
<td>30404</td>
<td></td>
</tr>
<tr>
<td>31533</td>
<td>31538</td>
<td>31338</td>
<td>31554</td>
<td>31012</td>
<td>31459</td>
<td>31783</td>
<td>31077</td>
<td>31827</td>
<td>31349</td>
<td>31457</td>
<td>31203</td>
<td>31077</td>
<td>30828</td>
<td></td>
</tr>
<tr>
<td>31469</td>
<td>31294</td>
<td>31186</td>
<td>31286</td>
<td>30812</td>
<td>31215</td>
<td>31591</td>
<td>31084</td>
<td>31588</td>
<td>31216</td>
<td>31246</td>
<td>30964</td>
<td>30717</td>
<td>30865</td>
<td></td>
</tr>
<tr>
<td>30603</td>
<td>30609</td>
<td>30406</td>
<td>30676</td>
<td>30357</td>
<td>30472</td>
<td>30717</td>
<td>30355</td>
<td>30907</td>
<td>30736</td>
<td>30713</td>
<td>30328</td>
<td>30294</td>
<td>30085</td>
<td></td>
</tr>
<tr>
<td>32188</td>
<td>32136</td>
<td>31977</td>
<td>32140</td>
<td>31647</td>
<td>31975</td>
<td>32041</td>
<td>31445</td>
<td>32717</td>
<td>31847</td>
<td>31578</td>
<td>31257</td>
<td>31575</td>
<td></td>
<td></td>
</tr>
<tr>
<td>31149</td>
<td>31059</td>
<td>30705</td>
<td>30995</td>
<td>30550</td>
<td>30944</td>
<td>31182</td>
<td>30654</td>
<td>31344</td>
<td>31015</td>
<td>31132</td>
<td>30825</td>
<td>30727</td>
<td>30331</td>
<td></td>
</tr>
<tr>
<td>31831</td>
<td>31776</td>
<td>31455</td>
<td>31789</td>
<td>31219</td>
<td>31752</td>
<td>32097</td>
<td>31453</td>
<td>31997</td>
<td>31873</td>
<td>31944</td>
<td>31655</td>
<td>31554</td>
<td>31193</td>
<td></td>
</tr>
<tr>
<td>32638</td>
<td>32608</td>
<td>32338</td>
<td>32712</td>
<td>32032</td>
<td>32407</td>
<td>32635</td>
<td>32325</td>
<td>32845</td>
<td>32647</td>
<td>32752</td>
<td>32314</td>
<td>32307</td>
<td>31921</td>
<td></td>
</tr>
<tr>
<td>32213</td>
<td>33105</td>
<td>32872</td>
<td>33247</td>
<td>32431</td>
<td>33063</td>
<td>33174</td>
<td>32579</td>
<td>33577</td>
<td>32977</td>
<td>32882</td>
<td>32754</td>
<td>32391</td>
<td>32650</td>
<td></td>
</tr>
<tr>
<td>33015</td>
<td>33109</td>
<td>32746</td>
<td>33167</td>
<td>32508</td>
<td>32889</td>
<td>33187</td>
<td>32666</td>
<td>33140</td>
<td>33107</td>
<td>33178</td>
<td>32890</td>
<td>32745</td>
<td>32492</td>
<td></td>
</tr>
<tr>
<td>33211</td>
<td>33225</td>
<td>33069</td>
<td>33266</td>
<td>32846</td>
<td>32830</td>
<td>32895</td>
<td>32549</td>
<td>32222</td>
<td>32944</td>
<td>32865</td>
<td>32476</td>
<td>32307</td>
<td>32206</td>
<td></td>
</tr>
<tr>
<td>34072</td>
<td>33869</td>
<td>33875</td>
<td>33230</td>
<td>33274</td>
<td>33590</td>
<td>32823</td>
<td>33583</td>
<td>32773</td>
<td>32796</td>
<td>32665</td>
<td>32239</td>
<td>32136</td>
<td></td>
<td></td>
</tr>
<tr>
<td>33857</td>
<td>33437</td>
<td>33225</td>
<td>33299</td>
<td>33586</td>
<td>33760</td>
<td>33061</td>
<td>33278</td>
<td>33144</td>
<td>32759</td>
<td>32546</td>
<td>32396</td>
<td>32226</td>
<td>32195</td>
<td></td>
</tr>
<tr>
<td>33032</td>
<td>33008</td>
<td>33239</td>
<td>33299</td>
<td>32869</td>
<td>33061</td>
<td>33278</td>
<td>33144</td>
<td>32759</td>
<td>32546</td>
<td>32396</td>
<td>32226</td>
<td>32195</td>
<td></td>
<td></td>
</tr>
<tr>
<td>33504</td>
<td>33366</td>
<td>33274</td>
<td>33586</td>
<td>32869</td>
<td>33200</td>
<td>32954</td>
<td>33947</td>
<td>33129</td>
<td>33185</td>
<td>32938</td>
<td>32821</td>
<td>32435</td>
<td></td>
<td></td>
</tr>
<tr>
<td>33880</td>
<td>33585</td>
<td>33590</td>
<td>33760</td>
<td>33061</td>
<td>33200</td>
<td>32954</td>
<td>33947</td>
<td>33129</td>
<td>33185</td>
<td>32938</td>
<td>32821</td>
<td>32435</td>
<td></td>
<td></td>
</tr>
<tr>
<td>33297</td>
<td>32893</td>
<td>32823</td>
<td>33064</td>
<td>32578</td>
<td>32954</td>
<td>33360</td>
<td>33220</td>
<td>32988</td>
<td>32973</td>
<td>32718</td>
<td>32602</td>
<td>32029</td>
<td>32772</td>
<td></td>
</tr>
<tr>
<td>33767</td>
<td>33643</td>
<td>33583</td>
<td>33870</td>
<td>33144</td>
<td>33947</td>
<td>34109</td>
<td>33220</td>
<td>32988</td>
<td>32973</td>
<td>32718</td>
<td>32602</td>
<td>32029</td>
<td>32772</td>
<td></td>
</tr>
<tr>
<td>33047</td>
<td>32975</td>
<td>32773</td>
<td>33266</td>
<td>32759</td>
<td>33129</td>
<td>33510</td>
<td>32988</td>
<td>33606</td>
<td>33302</td>
<td>32970</td>
<td>32973</td>
<td>32063</td>
<td>32818</td>
<td></td>
</tr>
<tr>
<td>33195</td>
<td>33040</td>
<td>32976</td>
<td>33237</td>
<td>32546</td>
<td>33185</td>
<td>33579</td>
<td>32973</td>
<td>33583</td>
<td>33302</td>
<td>33085</td>
<td>33063</td>
<td>32681</td>
<td>32302</td>
<td></td>
</tr>
<tr>
<td>33243</td>
<td>32796</td>
<td>32665</td>
<td>3309</td>
<td>32396</td>
<td>32938</td>
<td>33556</td>
<td>32718</td>
<td>33355</td>
<td>32970</td>
<td>33058</td>
<td>32818</td>
<td>32446</td>
<td>32302</td>
<td></td>
</tr>
</tbody>
</table>

Table 2: Pair-wise Vertex Intersections, [1979, 2005] × [1992, 2005]
Table 3: Global Intersection Number and Number of Vertices

<table>
<thead>
<tr>
<th>$i$</th>
<th>$NGIN(i)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1011</td>
</tr>
<tr>
<td>3</td>
<td>1310</td>
</tr>
<tr>
<td>4</td>
<td>1568</td>
</tr>
<tr>
<td>5</td>
<td>1335</td>
</tr>
<tr>
<td>6</td>
<td>1057</td>
</tr>
<tr>
<td>7</td>
<td>858</td>
</tr>
<tr>
<td>8</td>
<td>748</td>
</tr>
<tr>
<td>9</td>
<td>574</td>
</tr>
<tr>
<td>10</td>
<td>504</td>
</tr>
<tr>
<td>11</td>
<td>420</td>
</tr>
<tr>
<td>12</td>
<td>379</td>
</tr>
<tr>
<td>13</td>
<td>335</td>
</tr>
<tr>
<td>14</td>
<td>360</td>
</tr>
<tr>
<td>15</td>
<td>346</td>
</tr>
<tr>
<td>16</td>
<td>368</td>
</tr>
<tr>
<td>17</td>
<td>445</td>
</tr>
<tr>
<td>18</td>
<td>579</td>
</tr>
<tr>
<td>19</td>
<td>653</td>
</tr>
<tr>
<td>20</td>
<td>811</td>
</tr>
<tr>
<td>21</td>
<td>988</td>
</tr>
<tr>
<td>22</td>
<td>1200</td>
</tr>
<tr>
<td>23</td>
<td>1592</td>
</tr>
<tr>
<td>24</td>
<td>2108</td>
</tr>
<tr>
<td>25</td>
<td>2689</td>
</tr>
<tr>
<td>26</td>
<td>5214</td>
</tr>
<tr>
<td>27</td>
<td>18511</td>
</tr>
</tbody>
</table>

Figure 7: $NGIN(i)$ for $2 \leq i \leq 27$
Figure 8: 4 Vertex Intersections with Map Overlay

Circle are white, indicating that they are involved in all of the largest connected components of all years. This is interesting because the property of these points is that they are central to all of the graphs’ structure, and are important for analyzing in terms of their concentration. What is also interesting is that the Labrador Sea (southwest corner of Greenland) also has many points, indicating that this region is also important for study, as was also shown in Section 2.
Figure 9: 10 Vertex Intersections with Map Overlay

Figure 10: 25 Vertex Intersections with Map Overlay
Figure 11: 27 Vertex Intersections with Map Overlay
# 4 Betweenness Centrality

## 4.1 Betweenness Centrality

Betweenness centrality is a measure of a particular node’s centrality in a graph. Specifically, for a node $v$, its betweenness centrality value is defined as:

\[
BetweennessCentrality(v) = \sum_{s \neq v \neq t} \frac{\sigma_{st}(v)}{\sigma_{st}},
\]

where $\sigma_{st}$ is the number of shortest paths from $s$ to $t$, and $\sigma_{st}(v)$ are those that pass through $v$. Let the quantity $\frac{\sigma_{st}(v)}{\sigma_{st}} = 0$ if $s$ and $t$ do not have a path through $v$, or are not in the same connected component.

Examples of low and high betweenness values are the following: suppose there are two large, separate clusters of vertices, and only one vertex connected by one edge in between them. This vertex has high betweenness value because the ratio of the number of shortest paths that pass through it to the number of shortest paths between any two vertices in the clusters is high. However, if there is a cluster of vertices and one vertex is only connected by one edge to one other vertex in the cluster, it will have a low betweenness value because very few (if any) shortest paths pass through that vertex.

The fastest known serial algorithm for exact betweenness centrality is defined in [9]; however, the algorithm takes a long time to run for large data sets: for any single vertex $v$, calculating its betweenness value takes $O(|V| \times |E|)$ time, where $G = (V, E)$ is an undirected graph. Therefore, calculating betweenness centrality values for all vertices $v \in V$ takes $O(|V|^2 \times |E|)$ time.

With many tens of thousands of vertices in these graphs, the algorithm takes a long time to finish. Therefore, we need to find a way to speed up the algorithm. Since the graph depends on the number of shortest paths between a node, the results for each connected component are independent of each other. The reasoning is as follows: let $G = (V, E)$ be an undirected graph, $G_1 = (V_1, E_1), ..., G_k = (V_k, E_k)$ be its connected components. From the definition of betweenness centrality, for a vertex $v \in V$,

\[
BC(v) = \sum_{s, t \in V} \frac{\sigma_{st}(v)}{\sigma_{st}}.
\]

Consider a vertex $v \in V_m, 1 \leq m \leq k$. Since $G$ is undirected, $V_i \cap V_j = \emptyset$ for $i \neq j$. Therefore, we can see that:

\[
BC(v) = \sum_{s, t \in V} \frac{\sigma_{st}(v)}{\sigma_{st}}
= \sum_{s, t \in V_1} \frac{\sigma_{st}(v)}{\sigma_{st}} + \ldots + \sum_{s, t \in V_k} \frac{\sigma_{st}(v)}{\sigma_{st}} + 2 \sum_{s \in V_i, t \in V_j, i \neq j} \frac{\sigma_{st}(v)}{\sigma_{st}}
= 0 + \ldots + 0 + \sum_{s, t \in V_m} \frac{\sigma_{st}(v)}{\sigma_{st}} + 0 + \ldots + 0
= \sum_{s, t \in V_m} \frac{\sigma_{st}(v)}{\sigma_{st}}.
\]
Since \( V_i \cap V_j = \emptyset \) for \( i \neq j \), this algorithm is highly parallelizable; see Section 7.3 for details. We generate images visualizing betweenness centrality values, as follows:

- Run the Betweenness Centrality algorithm on all \( v \in V \), store result into a list \( L \).
- For each \( l \in L \), scale the value to an integer between 0 and 255, and record \( l \)'s x and y coordinates.
- For each of the new integer values \( i \), set \( i_2 = i << 16 | i << 8 | i \).
- If there is aggregation \( d \), \( \forall x', y' \) such that \( x \leq x' \leq x + d \) and \( y \leq y' \leq y + d \), draw \( i_2 \) at coordinate \( (x', y') \). Otherwise (i.e., \( d = 1 \)), draw \( i_2 \) at coordinate \( (x, y) \). The drawing of the point corresponds to the value between 0 and 255. If the value is 0, the pixel drawn is black; likewise, for 255, it is white. Any other value is a linear progression from black to white (i.e., 128 is gray).
- Apply the map overlay provided by GSFC to the images to show the land masses (dark blue).

An intuition for the image generation algorithm is as follows: we take all of the betweenness values of the graph, and compute the logarithm of each value (the purpose of the logarithm is to make the results more visually clear). Then, we map the resulting values to a linear scale between 0 and 255. The higher a particular vertex’s betweenness value is, the higher is the mapped value. In Figure 13 the brighter (i.e. white) vertices are those that have a high mapped value, and the darker (i.e., black) vertices are those that have a low mapped value. The intuition in the graph is to make visible the locations of vertices with high betweenness, and to see if there are any trends with vertices with low betweenness.

### 4.2 Betweenness Centrality Results

We ran standard algorithms for betweenness (along with an improvement in performance; see Section 7.3) and output data for each of the years (in the `betweenness_centrality` directory):

- `<year>.betweenness`, `<year>MaxSet.betweenness`
- `<year>.png`, `<year>MaxSet.png`
- `output<year>.txt`, `output<year>MaxSet.txt`

The .png files correspond to the images that are generated, such as in Figures 13 and 14. The .betweenness files contain each of the vertices of the graph along with their betweenness values. The .txt files contain a sorted list of the betweenness values for each of the years.

In Figure 13 we show the result of this algorithm for the year 1983 (there are similar results for all other years). However, since the betweenness values for these graphs follow an exponential curve (i.e., few vertices have high betweenness values, and many have low values), we took the logarithm of each of the values, and ran the algorithm with these new values. We plot this result in Figure 12 where the horizontal axis represents the vertex identifier, and the vertical axis is its corresponding betweenness value (in sorted increasing order). Since these graphs resemble small-world networks, we can reasonably conclude that small-world graphs have this pattern with betweenness values, as shown through various examples in [6].
We also performed the same process but instead with the largest connected component. We should not see much of a noticeable difference in the image generated, since the definition of betweenness centrality deals with a sum of $n^2$ pairs of vertices in a connected component of size $n$. However, there is quite a difference. For example, observe the lower-left corner of Figures 13 and 14 in Figure 13 there are various “streaks” of points, which indicate that all of those physically adjacent areas had high betweenness value. However, in Figure 14 they are not visible. Another example is near the southeast corner of Greenland: in Figure 13 there is a cluster of grey points, which indicate a betweenness value that is near the median value. However, in Figure 14 this cluster does not appear.

An important part of analyzing centrality measures is to have an upper limit on the values. In Table 4 and Figure 15 we give the maximal betweenness values for each year. There does not seem to be any pattern over time with regard to these maximal values. However, there are interesting results when we consider every single vertex’s betweenness value. For example, the year 1991 has a betweenness value much larger than any other year (1872647). However, none of the other plots and graphs throughout this thesis exhibit any reason as to why this year has such a large betweenness value.

Looking at the file output1991.txt, one would find how the next largest betweenness values drop off very quickly from the maximum, whereas some other years with a lower maximal betweenness value have a slower rate of decrease. This analysis indicates that the graphs with more uniform distributions of betweenness are more centralized, in that there are fewer isolated clusters of vertices.
Figure 13: Betweenness Centrality values, 1983

Figure 14: Betweenness Centrality values, Max Set, 1983
Table 4: Max Betweenness values per year

<table>
<thead>
<tr>
<th>Year</th>
<th>Max Betweenness</th>
</tr>
</thead>
<tbody>
<tr>
<td>1979</td>
<td>235035</td>
</tr>
<tr>
<td>1980</td>
<td>18415</td>
</tr>
<tr>
<td>1981</td>
<td>10193</td>
</tr>
<tr>
<td>1982</td>
<td>112848</td>
</tr>
<tr>
<td>1983</td>
<td>8314</td>
</tr>
<tr>
<td>1984</td>
<td>742191</td>
</tr>
<tr>
<td>1985</td>
<td>29975</td>
</tr>
<tr>
<td>1986</td>
<td>22187</td>
</tr>
<tr>
<td>1987</td>
<td>11770</td>
</tr>
<tr>
<td>1988</td>
<td>15034</td>
</tr>
<tr>
<td>1989</td>
<td>29563</td>
</tr>
<tr>
<td>1990</td>
<td>7016</td>
</tr>
<tr>
<td>1991</td>
<td>1872647</td>
</tr>
<tr>
<td>1992</td>
<td>91240</td>
</tr>
<tr>
<td>1993</td>
<td>9884</td>
</tr>
<tr>
<td>1994</td>
<td>7359</td>
</tr>
<tr>
<td>1995</td>
<td>123701</td>
</tr>
<tr>
<td>1996</td>
<td>5014</td>
</tr>
<tr>
<td>1997</td>
<td>12714</td>
</tr>
<tr>
<td>1998</td>
<td>74347</td>
</tr>
<tr>
<td>1999</td>
<td>60921</td>
</tr>
<tr>
<td>2000</td>
<td>19938</td>
</tr>
<tr>
<td>2001</td>
<td>167325</td>
</tr>
<tr>
<td>2002</td>
<td>37922</td>
</tr>
<tr>
<td>2003</td>
<td>4317</td>
</tr>
<tr>
<td>2004</td>
<td>19527</td>
</tr>
<tr>
<td>2005</td>
<td>92825</td>
</tr>
</tbody>
</table>

that are connected by few edges.
Figure 15: Max Betweenness values per year


5 Induced Time Lag

Another way to investigate shifts in the behavior of arctic ice is to investigate a time lag when computing a correlation-based graph. Since the data is broken up into weeks of data, we investigate a time lag of 1, 2, 3, and 4 weeks. The motivation for this is to see if predictable patterns happen earlier in the year over time, such as the seasonal melting of ice, through some measurable value. A result of the ice’s rate of melting earlier in the year (and consequently, re-freezing occurring later in the year) over time would mean that there is less seasonal ice coverage. Our goals are to:

- Choose an offset $s \in \{1, 2, 3, 4\}$ (or if desired, $s \in \{-1, -2, -3, -4\}$ to represent a “back-shift”).
- Build graphs in the same manner as was done in Section 1.2, but instead:
  - Choose a set of weeks $w_1, ..., w_n \subset [1, 52]$.
  - Build 2 graphs: one for the weeks $w_1, ..., w_n$, and the other for weeks $w_1 + s, ..., w_n + s$.
- Repeat for all values of $s$.
- With all of these graphs, make measurements on the differences in ice concentrations.

5.1 Vertex Similarity

We need a metric to relate graphs, and especially graphs of different sizes. The Graph Isomorphism (GI) problem relates graphs that can be mapped 1-1 with each other; more precisely,

**Definition** Given two graphs $G = (V_G, E_G)$ and $H = (V_H, E_H)$, and a function $f : V_G \rightarrow V_H$. There is an isomorphism between $G$ and $H$ (equivalently, they are isomorphic) only when the following is true: $\forall u, v \in V_G, \{u, v\} \in E_G$ if and only if $\{f(u), f(v)\} \in E_H$.

However, this definition is not very helpful for two reasons. First, the GI problem is not even known to be in $\mathcal{P}$ [13], and our goal is to find more tractable algorithms. Also, the GI problem does not give good intuition if two graphs are not exactly isomorphic. Another metric that does have this intuition is a vertex similarity approximation algorithm given in Blondel et al.[8, 7], given in Algorithm 2, where the normalization is done column-wise by sum of the column.

**Algorithm 2: Vertex Similarity**

The result of the algorithm is as follows: the $ij$-th entry in the output matrix is a value between 0 and 1 describing how much vertex $i \in V_G$ is similar to vertex $j \in V_H$ in terms of neighboring vertices. A value closer to 0 describes that the vertices do not share the same neighbor vertices, whereas they do if the value is closer to 1.
5.2 Cosine Similarity

Though Blondel et al.’s similarity metric is useful (and also the improvements using the Hungarian assignment algorithm in [7]), it is impractical for graphs of the size in this thesis. Therefore, we turn to cosine similarity as a metric for vertex/graph similarity, as it is a more tractable algorithm as a similarity metric.

Cosine similarity is a metric commonly used in Linear Algebra to denote a value of similarity between two vectors in an inner product space. The cosine of an angle $\theta$ is related to the dot product of two vectors $a, b$ in that space by their magnitudes. More formally,

$$a \cdot b = ||a|| ||b|| \cos(\theta).$$

Another reason we look to cosine similarity is that it takes into account varying degrees of vertices and how many common neighbors are shared between each pair of vertices. The algorithm is:

1. From two input graphs $G_1, G_2$, let $x$ and $y$ be arbitrary vertices of these graphs, respectively.
2. Let $A_x, A_y$ be lists of integers all initially set to 0, where all elements in each correspond to vertices of $G_1, G_2$, respectively, in some order (As long as the order is fixed throughout the calculations, the order does not matter.). $A_x[i]$ corresponds to whether $x$ and the corresponding vertex of $A_x[i]$ are connected or not; if so, then set $A_x[i] = 1$, and 0 otherwise ($A_y[i]$ is defined similarly).
3. Compute:

$$\frac{A_x \cdot A_y}{||A_x|| ||A_y||}$$

where $A_x \cdot A_y$ is the dot product of $A_x$ and $A_y$, and $||A_x||$ is the magnitude of $A_x$. This value is the similarity metric for vertices $x$ and $y$. A value of 1 indicates that they have exactly the same neighbors, 0 means that they do not.

We now discuss the values for cosine similarity for Watts-Strogatz graphs, and compare them to the graphs studied in this thesis. Watts-Strogatz graphs have small-world properties, and so we derive expected cosine similarity values for them, using random graphs. A random graph is defined as $G^R = (V^R, E^R)$ with a parameter $0 < p < 1$ such that $\forall v_i, v_j \in V^R, (v_i, v_j) \in E^R$ with probability $p$ [12].

**Theorem 5.1.** For two random graphs $G_1^R = (V_1^R, E_1^R)$ and $G_2^R = (V_2^R, E_2^R)$ with probabilities $p_1, p_2$, respectively, and where $n = |V_1^R| = |V_2^R|$, the expected average cosine similarity value for them is $\sqrt{p_1 p_2}$.

**Proof.** Take any two vertices $v_1 \in V_1^R, v_2 \in V_2^R$ and their associated adjacency lists $a_1 = [b_{11}, ..., b_{n1}], a_2 = [b_{12}, ..., b_{n2}]$ where all of the $b_{1i}, b_{2i} \in \{0, 1\}$. In the summation below (the dot product of the two lists) each $b_{1i}b_{2i} = 1$ if and only if both values are 1, which occurs with probability $p_1 p_2$:

$$\sum_{i=1}^{n} b_{1i}b_{2i}.$$
Induced Time Lag

Therefore, the expected value of this summation is $p_1 p_2 n$. The denominator of the cosine similarity value is the product of the magnitudes of these two adjacency lists, i.e.,

$$\sqrt{\sum_{i=1}^{n} (b_{i1})^2} \sqrt{\sum_{i=1}^{n} (b_{i2})^2}.$$

Now, since $b_{i1}, b_{i2} \in \{0, 1\}$, this is equal to:

$$\sqrt{\sum_{i=1}^{n} b_{i1}} \sqrt{\sum_{i=1}^{n} b_{i2}}.$$

As we have seen, these sums are equal to $p_1 n$ and $p_2 n$, respectively:

$$\sqrt{p_1 n} \sqrt{p_2 n} = n \sqrt{p_1 p_2}.$$

Therefore, the quotient of these two values is:

$$\frac{np_1 p_2}{n \sqrt{p_1 p_2}} = \sqrt{p_1 p_2}.$$

In [16], statistical comparisons concluded that the graphs described by the NSIDC data set are similar to the Watts-Strogatz model of small-world graphs [22], in that they are constructed with average vertex degree $k$, and setting the edge probability to be $\frac{nk}{n(n-1)}$. From this, we get the following theorem:

**Theorem 5.2.** The expected average cosine similarity value for a Watts-Strogatz model for two graphs of $n$ vertices with average degree $k_1, k_2$ respectively is $\frac{1}{n-1} \sqrt{k_1 k_2}$.

**Proof.** Substituting $p_i = \frac{nk_i}{n(n-1)}$ for $i \in \{1, 2\}$ into the result of Theorem [5.1] gives:

$$\sqrt{p_1 p_2} = \sqrt{\frac{n^2 k_1 k_2}{(n(n-1))^2}}$$

$$= \sqrt{\frac{n^2 k_1 k_2}{n^2(n-1)^2}}$$

$$= \sqrt{\frac{k_1 k_2}{(n-1)^2}}$$

$$= \frac{1}{n-1} \sqrt{k_1 k_2}$$

$$5.3 \text{ Cosine Similarity Results}$$

All of the theory and analysis in Theorems [5.1] and [5.2] allow us to understand what the expected results of the class of graphs that the NSIDC data sets represent, and how they relate to what the
actual results are. They allow us to see what differences that are generated not only by comparisons between the yearly graphs, but also the weekly ones with offsets that we constructed.

Even though cosine similarity is far more tractable than Blondel et al.’s metric, it is still too expensive of a computation to find the similarity between all billions of pairs of vertices between the input graphs. Therefore, we developed a “randomized” approach: consider $n$ randomly chosen vertices in both graphs, and calculate the values for all $n^2$ distinct pairs of vertices. Then, repeat this algorithm $m$ times, and find the average of all $m$ runs. For our tests, we chose $n = m = 100$ because these values allow us to sample many pairs of vertices.

First, we compare the graphs that have been studied throughout this thesis - namely, the graphs that correspond to each of the years in 1979 to 2005. The method of comparison was to look at for every year $i$, compare it to $i + 1$. In Figures 16 and 17, we plot the maximum and average cosine similarity. On the horizontal axis, we plot the year compared to the next, and the vertical axis has the maximum/average cosine similarity of all tested pairs of vertices.

We can see that from these values that the average cosine similarity has low value. Using the average degree of the two tested graphs and plugging them in to the results of Theorem 5.2 gives very similar results in terms of numerical value, as shown by Figure 17. For every year except 1994, the actual average value is larger (and sometimes much larger) than the calculated average for a graph with the same vertex degree. What is interesting is that for the last few years, the gap between the actual and calculated average values decreases, and comparing the years 2004 and 2005 have nearly identical values. At no other points in Figure 17 does this pattern happen.

Now we examine for induced time lag. Since there are so many different possible subsets of weeks and years that can be chosen for calculating cosine similarity values, we only choose a few. We chose to build our weekly graphs with a time period of 4 weeks each, and constructed nearly equal offsets of years in the data set; namely, we construct a graph of the year 1979 of weeks 1-4, 1988 of weeks 2-5, 1997 of weeks 3-6, and 2005 of weeks 4-7. These years were chosen to give a reasonable distribution of the years, and to capture any “long-term” trends. From these 4 graphs,
we compare the average and maximum cosine similarity values between all 6 pairs of graphs.

In Table 6, we provide each of the graphs along with the average cosine similarity between each pair of graphs (and correspondingly the maximum in Table 5). Most of the values are within the range of values that are given in Figures 16 and 17. However, comparing the 1979 weeks 1-4 graph with the 2005 4-7 graph give a much larger value than any of the other average cosine similarity values. Another interesting part about Table 6 is that the values increase the larger the gap in-between the tested years. For example, this is to suggest that the graph formed by weeks 1-4 of 1979 has a very similar structure with weeks 4-7 of 2005. Even though all of the values in Table 6 are quite small, they allow us to see the trend where there is a higher correlation between larger gaps of weeks and years.

In Table 5, there are quite a few interesting pieces of data. First, the trend of increasing cosine similarity value as the difference in the years (and weeks) compared is the same as in Table 6. However, one can see that many of the values are quite large, especially comparing weeks 1-4 of 1979 and weeks 4-7 of 2005 (a value of 0.791). As seen in Figure 16, the maximum value achieved was in the year 1991 with a value of 0.622. This evidence shows that offsetting data by weeks over different years has higher correlation than either calculated or even actual results of only comparing two graphs that represent whole years. Therefore, when looking at the sea ice data, we can conclude that the concentrations over long-term time intervals are more correlated than short-term intervals.
<table>
<thead>
<tr>
<th></th>
<th>1979 (1-4)</th>
<th>1988 (2-5)</th>
<th>1997 (3-6)</th>
<th>2005 (4-7)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1979 (1-4)</td>
<td>—</td>
<td>0.563</td>
<td>0.692</td>
<td>0.791</td>
</tr>
<tr>
<td>1988 (2-5)</td>
<td>0.563</td>
<td>—</td>
<td>0.557</td>
<td>0.638</td>
</tr>
<tr>
<td>1997 (3-6)</td>
<td>0.692</td>
<td>0.557</td>
<td>—</td>
<td>0.426</td>
</tr>
<tr>
<td>2005 (4-7)</td>
<td>0.791</td>
<td>0.638</td>
<td>0.426</td>
<td>—</td>
</tr>
</tbody>
</table>

Table 5: 4-Week Maximum Cosine Similarity Values

<table>
<thead>
<tr>
<th></th>
<th>1979 (1-4)</th>
<th>1988 (2-5)</th>
<th>1997 (3-6)</th>
<th>2005 (4-7)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1979 (1-4)</td>
<td>—</td>
<td>0.0193</td>
<td>0.0284</td>
<td>0.0488</td>
</tr>
<tr>
<td>1988 (2-5)</td>
<td>0.0193</td>
<td>—</td>
<td>0.0233</td>
<td>0.0281</td>
</tr>
<tr>
<td>1997 (3-6)</td>
<td>0.0284</td>
<td>0.0233</td>
<td>—</td>
<td>0.0210</td>
</tr>
<tr>
<td>2005 (4-7)</td>
<td>0.0488</td>
<td>0.0281</td>
<td>0.0210</td>
<td>—</td>
</tr>
</tbody>
</table>

Table 6: 4-Week Average Cosine Similarity Values
6 Characteristic Path Length and Connected Components

We turn to a different method of analyzing the graphs, mainly in terms of how connected they are: characteristic path length (CPL). The analysis of centrality and contours does give various approaches in understanding the graphs’ structure in relation to other classes of graphs, but the CPL can also characterize graphs in terms of how “tightly” connected they are. The CPL of a graph $G$ is the average of the length of the shortest path between any pair of vertices of $G$. Specifically, for a graph $G = (V, E)$, the CPL of $G$ is:

$$\text{CPL}_G = \frac{2 \sum_{i \neq j} d(v_i, v_j)}{|V| \times (|V| - 1)}$$

for $v_i, v_j \in V, d(v_i, v_j)$ is the shortest path connecting $v_i, v_j$ (not included in calculation if no path exists). The motivation for studying the CPL of a graph in general is that it relates to how quickly information can flow through the graph, as well as how connected the graphs are. For example, if a graph has a low CPL, then the average length of the paths between all pairs of vertices is low, implying that information can reach the destination quickly on average.

Computing the characteristic path length (CPL) of the entire arctic ice data set is expensive. Even evaluating the CPL for one year takes many hours to compute and many gigabytes of RAM. Floyd-Warshall’s all-pairs shortest path algorithm (APSP) was implemented in [16], which has a running time of $\Theta(n^3)$, where $n = |V_G|$. Also, it remains an open question of whether APSP can be solved in $O(n^3 - \epsilon)$ time for some $\epsilon > 0$ [2, 4]. A recent paper [24] describes how, if “additions and comparisons of reals are unit cost (but all other operations have typical logarithmic cost),” the algorithm for APSP runs in time $\frac{n^3}{2^{\Omega((\log n)^2)}}$. Therefore, we compute the CPL on a subset of the data: namely, each of the calculated connected components. For Theorem 6.1, we define an “operation” as one of the following:

- Given two integers $i, j$, retrieving/storing an integer in a matrix at position $(i, j)$.
- Addition of two integers.

**Theorem 6.1.** Let $G = (V, E)$ be a simple, undirected graph, and $k$ the number of connected components in $G$. Let $n = |V|$. If $k = 1$, computing the CPL of all connected components takes $2n^3$ operations. If $k > 1$, this computation takes strictly fewer than $2n^3$ operations.

**Proof.** Let:

- $G = (V, E)$ be an simple, undirected graph
- $C_1 = (V_1, E_1), ..., C_k = (V_k, E_k)$ be the connected components of $G$
- $n_1 = |V_1|, ..., n_k = |V_k|$
- $\text{CPL}_i$ be the characteristic path length of $C_i$.

An analysis of Floyd-Warshall’s all-pairs shortest path algorithm reveals that for a graph with $n$ vertices, computing the characteristic path length of that graph takes $2n^3$ operations (minus preprocessing). Likewise, computing $\text{CPL}_i$ takes $2n_i^3$ operations. Let $c_i = \frac{n_i}{n}$. Therefore, computing $\text{CPL}_i$ takes $2(2c_i^3) = 2n_i^3 \frac{c_i^3}{c_i}$ operations. We can see that computing the CPL of $G$ (i.e., of all connected
Characteristic Path Length and Connected Components

components) takes \(2n^3 \sum_{i=1}^{k} \frac{1}{c_i^3}\) operations (just the sum over all connected components). We can separate the results of each connected component since the graph is undirected (i.e., \(V_i \cap V_j = \emptyset, \forall i \neq j\)). We can see that \(c_i > 1\) if \(k > 1\), and \(c_i = 1\) if \(k = 1\). Consider the former case. Therefore,

\[
\sum_{i=1}^{k} \frac{1}{c_i} = \sum_{i=1}^{k} \frac{n_i}{n} = 1
\]

and:

\[
1 = \sum_{i=1}^{k} \frac{1}{c_i} = \frac{1}{c_1} + ... + \frac{1}{c_k} > \frac{1}{c_1^3} + \frac{1}{c_2^3} + ... + \frac{1}{c_k^3} = \sum_{i=1}^{k} \frac{1}{c_i^3}.
\]

Since \(\sum_{i=1}^{k} \frac{1}{c_i^3} < 1\) for \(k > 1\), \(2n^3 \sum_{i=1}^{k} \frac{1}{c_i^3} < 2n^3\). Substituting \(k = 1\) yields \(2n^3\) operations, whereas \(k > 1\) yields fewer than \(2n^3\) operations, which concludes the proof.

We can further motivate this result by observing that if the number of vertices in each connected component is approximately the same, one can parallelize the algorithm more effectively (balanced workload per thread of execution). If this were the case, each connected component would have \(\approx \frac{n}{k}\) vertices, and the number of operations (after substituting and simplifying) would be \(2\frac{n^3}{k^2}\), a lower bound.

We can exploit the fact that the graphs are undirected when computing the CPL of the entire graph, by showing that the CPL of a simple, undirected graph and the CPLs of all of its connected components are related by:

\[
\text{CPL}_G = \sum_{i=1}^{k} \text{CPL}_i \times |V_i|\]  

if \(G = (V, E)\) is an undirected graph, \(CC_1 = (V_1, E_1), ..., CC_k = (V_k, E_k)\) are \(G\)'s connected components, and \(|V_1| + ... + |V_k| = |V|\). For a connected component \(CC_i = (V_i, E_i)\),

\[
\text{CPL}_i = \frac{2 \sum_{v_a, v_b \in V_i, v_a \neq v_b} d(v_a, v_b)}{|V_i|(|V_i| - 1)}
\]

by definition, where \(d(v_a, v_b)\) is the shortest distance between vertices \(v_a\) and \(v_b\). \(\text{CPL}_i\) is the average shortest distance between any pair of vertices in \(CC_i\). Here is another way to think of CPL: for a given vertex \(v\), the CPL for \(v\) of the average of the distances from \(v\) to all other vertices; and likewise, the CPL for an entire graph is the average of the CPL values for all vertices. We can reason that:

\[
\text{CPL}_i \times |V_i| = \frac{2 \sum_{v_a, v_b \in V_i, v_a \neq v_b} d(v_a, v_b)}{|V_i| - 1}
\]

is the total of all of the shortest distances between all pairs of vertices in \(CC_i\). Likewise, we can see that:

\[
\sum_{i=1}^{k} \text{CPL}_i \times |V_i|
\]

is the total of all of the shortest distances between all pairs of vertices in all connected components.

---

1 A proof of this result involves showing that for \(\forall k, 2k^3 < (k + 1)^3 + (k - 1)^3\).
of a graph $G = (V, E)$. Therefore, the average of the shortest distances of a graph $G$ is:

$$CPL_G = \frac{\sum_{i=1}^{k} CPL_i \times |V_i|}{|V|}.$$  

Computing one CPL of a particular year using Floyd-Warshall took over 2 days to complete on the A2C2 Advanced Computing Cluster \[3\]. Therefore, there was motivation to develop techniques for calculating the CPL using subsets of each data set. Indeed, using this analysis, the running time for computing the CPL of graphs was much lower (in fact, it was impossible to hold some of the adjacency matrices in memory); see Section 7.2.

Now we discuss improvements on our algorithm involving CC’s in computing the CPL of a graph. However, there are not many optimizations that can be done, as shown by Theorem 6.2.

**Theorem 6.2.** Let $G = (V, E)$ be any connected graph. Then,

$$\frac{2|V|(|V| - 1) - 2|E|}{|V|(|V| - 1)} \leq CPL_G \leq \frac{|V|^3 + 5|V| - 6|E| - 6}{3|V|(|V| - 1)}.$$

**Proof.** This is proven in [18], and is reproduced with slight modifications, as the proof there deals with the Wiener index of the graph, and not the CPL.

For the first inequality, let $u, v \in V$. If $(u, v) \in E$, then $d(u, v) = 1$ - if not, then $d(u, v) \geq 2$. Therefore, we can see that there are exactly $|E|$ ordered pairs of vertices that have a distance between them of 1, and $(\frac{|V|^2}{2}) - |E|$ ordered pairs that have distance $\geq 2$. Therefore,

$$CPL_G = \frac{2\sum_{u,v \in V} d(u, v)}{|V|(|V| - 1)} \geq \frac{2(|E| + 2 \times (\frac{|V|(|V| - 1)}{2} - |E|))}{|V|(|V| - 1)} = \frac{2|V|(|V| - 1) - 2|E|}{|V|(|V| - 1)}.$$

Now we consider the second inequality, with a proof by induction, and by investigating acyclic graphs first (trees in this case, since the graphs are undirected). Let $D_G = \sum_{i \neq j} d(v_i, v_j)$ for a graph $G = (V, E)$. We can see that $CPL_G = \frac{D_G}{|V|(|V| - 1)}$. Think of $D_G$ as the “sum of all the shortest paths for all pairs of vertices”. If $|V| = 2$ and $|E| = 1$, then the left and right hand sides are equal.

So assume that all connected graphs with the condition $|V| = n > 2$ have the inequality true. Let $T_G = (V_T, E_T)$ be an acyclic graph with $|V_T| = n + 1$, and $v \in V_T$ is a leaf (i.e. the degree of $v$ is 1). Also, let $T'_G = (V'_T, E'_T)$ be the tree that is induced by $V_T - \{v\}$.

$$D_T = D_{T'} + \sum_{u \neq v \in V_T} d(u, v) \leq \frac{|V_T|^3 + 5|V_T| - 6}{6} - V_T + 1 + \frac{|V_T|(|V_T| + 1)}{2}$$

$\Leftrightarrow CPL_G \leq \frac{(|V_T| + 2)(|V_T| + 1)}{3(|V_T| - 1)}.$

Now we generalize to any connected graph, and assume (by induction) that the inequality holds

---

\[2\] $D_G$ is sometimes called the Wiener index of $G$. 

Page 35 of 53
for any connected graph \( G = (V, E) \) with \( |E| \geq |V| - 1 \). Consider a graph \( G' = (V', E') \) with \( |V'| = |V| \) and \( |E'| = |E| + 1 \) with the same restriction. Since it is impossible for \( G' \) to be a tree, \( \exists e \in E' \) such that \( G'' = (V'', E'') \) is a connected subgraph of \( G' \), with \( V'' = V' \) and \( E'' = E' - \{e\} \). It is easily proven that \( D_{G'} \leq D_{G''} - 1 \). Therefore, by assumption, we have:

\[
D_{G'} \leq D_{G''} - 1 \leq \frac{|V|^3 + 5|V| - 6|E| - 6}{6} \Rightarrow CPL_{G} \leq \frac{|V|^3 + 5|V| - 6|E| - 6}{3|V|(|V| - 1)}
\]

Now, this result is for graphs with a fixed number of vertices and edges. If we allow any number of edges (so that the graph remains connected), we get the following theorem:

**Theorem 6.3.** Let \( G = (V, E) \) be any connected graph. Then,

\[
1 \leq CPL_{G} \leq \frac{|V| + 1}{3}.
\]

**Proof.** Substitute Theorem 6.2 with \( |E| = \frac{|V||V|-1}{2} \) for the first inequality, and \( |E| = |V| - 1 \) for the second. Even without using the theorem’s result, we can construct a graph with the two inequalities needed. The first one corresponds to a complete graph on \( |V| \) vertices. The second inequality corresponds to a graph of the following construction:

- Let \( G = (V, E) \) be a graph. Label the vertices in \( V \) as \( v_1, ..., v_k \) where \( k = |V| \).
- Edges are constructed as \((v_i, v_{i+1})\) for \( 1 \leq i \leq k - 1 \).

A calculation of the CPL of \( G \) using this construction gives \( \frac{|V|+1}{3} \).

### 6.1 CPL Results

All of Theorems 6.1 to 6.3 are for improving the algorithms to make them more tractable for graphs of this size. Even with all of these advancements in calculating CPL values, completion of the algorithm takes over 2 days, even only with one graph. In Figure 18, we give the year of the graph and corresponding CPL value.

What is more interesting is looking at how connected a graph is through the CPL metric. We now show the results of comparing the actual CPL results versus the maximum CPL value for graphs with the same number of vertices and edges. Specifically, we compare the values listed in Figure 18 versus the value below:

\[
\frac{1}{|V|} \sum_{i=1}^{k} \frac{|V_i|^3 + 5|V_i| - 3|E_i| - 6}{3(|V_i| - 1)}
\]

for a graph \( G = (V, E) \) with connected components \( CC_1 = (V_1, E_1), ..., CC_k = (V_k, E_k) \). In Figures 19 and 20, we give the same results as those in Figure 18 but also add the results of these calculations; for each year, the maximum CPL of a graph with the same number of vertices and edges as those of that year’s corresponding graph, and the quotient of the actual CPL and the maximum CPL.
Figure 18: CPL Results Per Year

Figure 19: Maximum CPL Per Year
Regarding Figure 19, the values are interesting, in that until around the year 1994, there is an upward trend from 6200 towards 7300, and after 1994, there is a downward trend back down to 6200. However, looking at Table 7, there does not seem to be anything special about the years in which the maximum CPL is larger than other years, since the calculation only involves a sum involving the number of vertices and edges in each component. Therefore, there must be a particular structure to those years that has them exhibit such a large maximum CPL value.

What is more interesting is that in Figure 20, there does not seem to be any effect by Figure 19, because the percentage (of the maximum CPL) values decrease over time, from 0.35 to 0.21. The plot is useful because the percentages are a measure of how connected the graph is only in terms of the number of vertices and edges it has. The more connected a graph is, the lower percentage it will have—the reason for this is that if a graph is highly connected, it will have a low CPL, and the maximal value will always be $O(|V|)$. Therefore, since the trend is that the ratio of actual CPL value to the maximal value decreases, we can reasonably conclude that the graphs, as well as the sea ice, become more connected over time.
7 Computational Advancements

Many of the algorithms that have been presented so far have high algorithmic costs; coupled with the very large data set provided, making optimizations to the algorithms is essential. Therefore, there is a motivation to understand various unique properties that the graphs have.

7.1 Graph Density

We now discuss the density of the graphs. A graph $G = (V,E)$ is called dense if $|E| \in \Theta(|V|^2)$, and sparse if $|E| \in O(|V|)$. Figure 21 and Table 7 give the year, number of vertices, and number of edges corresponding to that graph with no aggregation and threshold 0.9. We only include vertices if they are not isolated. As one can see, there is not only no trend in the number of edges over time, but also the relation to the number of vertices does not give a result as to the density or sparsity of the graphs. However, they do show that before 2003 (except for 1996), there are within 2.5 and 3.5 million edges in the graphs; and for years 2003-2005, the number of edges increases significantly. Therefore, we can conclude that in terms of the number of edges, most of the graphs are very similar, and that recent years have the graphs become more connected, as we have shown with CPL in Section 6.

7.2 Characteristic Path Length Running Times

We compare the running time of the original algorithm [16], and the run time using the results of Theorem 6.1 and the fact that:

$$CPL_G = \sum_{i=1}^{k} CPL_i \times |V_i| / |V|.$$  

In Table 8 and Figure 22 we document the year, old algorithm’s running time (in seconds), and the new algorithm’s running time (in seconds).

As one can see, the new algorithm is substantially faster in running time than the old algorithm for the same graph, and is so in each case by a speedup of 4-5 times. And, as is detailed before, the new algorithm is far more memory-efficient. Therefore, the new algorithm is preferable.

7.3 Parallelized Betweenness Centrality

We propose the following algorithm to parallelize betweenness centrality computations on undirected, unweighted graphs. This algorithm is adapted from [9]. Any change in the original algorithm is boxed.

We can guarantee correctness of this algorithm because we assume that $G$ is undirected and unweighted. This algorithm, as explained in Section 4.1, originally took $O(|V| \times |E|)$ time. Algorithm 3 executes in parallel on the graph’s connected components. Therefore, if $CC_1 = (V_1, E_1), ..., CC_k = (V_k, E_k)$ are a graph $G$’s connected components, and

$$n_i = max\{|V_i| \times |E_i|, \forall i \in [1,k], CC_i = (V_i, E_i)\}$$  

then this algorithm runs in $O(n_i)$ time. Also, the speedup of this algorithm is: $O\left(\frac{|V| \times |E|}{n_i}\right)$.
| Year | $|V|$    | $|E|$    |
|------|--------|--------|
| 1979 | 35013  | 3349800|
| 1980 | 33340  | 2115872|
| 1981 | 33410  | 2119600|
| 1982 | 34029  | 1930592|
| 1983 | 33900  | 2898416|
| 1984 | 32734  | 3259656|
| 1985 | 35265  | 2513678|
| 1986 | 33574  | 1909782|
| 1987 | 34336  | 2006280|
| 1988 | 34877  | 1612822|
| 1989 | 36031  | 2009904|
| 1990 | 35476  | 2005496|
| 1991 | 36116  | 2923204|
| 1992 | 37659  | 3074652|
| 1993 | 37828  | 2887328|
| 1994 | 37962  | 3025374|
| 1995 | 37823  | 3512560|
| 1996 | 36801  | 5714972|
| 1997 | 36717  | 2825374|
| 1998 | 37293  | 3166128|
| 1999 | 35794  | 3312152|
| 2000 | 37805  | 2924256|
| 2001 | 35884  | 2456250|
| 2002 | 35672  | 2324388|
| 2003 | 35657  | 2862030|
| 2004 | 35334  | 4057116|
| 2005 | 35029  | 5557852|

Table 7: Number of Vertices and Edges per Year

![Number of Edges](image)  
Figure 21: Number of Edges
<table>
<thead>
<tr>
<th>Year</th>
<th>Run Time (old)</th>
<th>Run Time (new)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1979</td>
<td>785156</td>
<td>164782</td>
</tr>
<tr>
<td>1980</td>
<td>583851</td>
<td>138967</td>
</tr>
<tr>
<td>1981</td>
<td>597428</td>
<td>142109</td>
</tr>
<tr>
<td>1982</td>
<td>651483</td>
<td>149653</td>
</tr>
<tr>
<td>1983</td>
<td>587736</td>
<td>146102</td>
</tr>
<tr>
<td>1984</td>
<td>629669</td>
<td>134418</td>
</tr>
<tr>
<td>1985</td>
<td>703308</td>
<td>173574</td>
</tr>
<tr>
<td>1986</td>
<td>645032</td>
<td>139957</td>
</tr>
<tr>
<td>1987</td>
<td>690577</td>
<td>166121</td>
</tr>
<tr>
<td>1988</td>
<td>777765</td>
<td>162507</td>
</tr>
<tr>
<td>1989</td>
<td>845498</td>
<td>177369</td>
</tr>
<tr>
<td>1990</td>
<td>713950</td>
<td>176316</td>
</tr>
<tr>
<td>1991</td>
<td>722166</td>
<td>175044</td>
</tr>
<tr>
<td>1992</td>
<td>859216</td>
<td>197144</td>
</tr>
<tr>
<td>1993</td>
<td>875016</td>
<td>202911</td>
</tr>
<tr>
<td>1994</td>
<td>1077688</td>
<td>229127</td>
</tr>
<tr>
<td>1995</td>
<td>948900</td>
<td>208123</td>
</tr>
<tr>
<td>1996</td>
<td>758356</td>
<td>184486</td>
</tr>
<tr>
<td>1997</td>
<td>808215</td>
<td>189408</td>
</tr>
<tr>
<td>1998</td>
<td>821695</td>
<td>203262</td>
</tr>
<tr>
<td>1999</td>
<td>863515</td>
<td>173328</td>
</tr>
<tr>
<td>2000</td>
<td>867173</td>
<td>204898</td>
</tr>
<tr>
<td>2001</td>
<td>744553</td>
<td>174460</td>
</tr>
<tr>
<td>2002</td>
<td>707453</td>
<td>166389</td>
</tr>
<tr>
<td>2003</td>
<td>767999</td>
<td>176825</td>
</tr>
<tr>
<td>2004</td>
<td>862744</td>
<td>175570</td>
</tr>
<tr>
<td>2005</td>
<td>701009</td>
<td>173717</td>
</tr>
</tbody>
</table>

Table 8: CPL Running Times, Old vs. New Algorithm

Figure 22: CPL Running Times, Old vs. New Algorithm
Computational Advancements

**Input:** $G = (V, E)$ undirected, unweighted graph

**Result:** $BC(v) =$ exact betweenness centrality value for $v \in V$

$C_B[v] = 0, \forall v \in V$;

$CC \leftarrow$ connected components of $G$;

for $CC_i = (V_i, E_i) \in CC$ in parallel do

for $s \in V_i$ do

$S \leftarrow$ empty stack;

$P[w] \leftarrow$ empty list, $w \in V_i$;

$\sigma[t] \leftarrow 0, t \in V_i$;

$\sigma[s] \leftarrow [1]$;

$d[t] \leftarrow -1, t \in V_i$;

$d[s] \leftarrow 0$;

$Q \leftarrow$ empty queue;

enqueue $s \rightarrow Q$;

while $Q$ not empty do

dequeue $v \leftarrow Q$;

push $v \rightarrow S$;

foreach $w \in v.neighbors()$ do

if $d[w] < 0$ then

enqueue $w \rightarrow Q$;

$d[w] \leftarrow d[v] + 1$;

end

if $d[w] = d[v] + 1$ then

$\sigma[w] \leftarrow \sigma[w] + \sigma[v]$;

append $v \rightarrow P[w]$;

end

end

$\delta[v] \leftarrow 0, v \in V_i$;

while $S$ not empty do

pop $w \leftarrow S$;

for $v \in P[w]$ do

$\delta[v] \leftarrow \delta[v] + \frac{\sigma[v]}{\sigma[w]} \times (1 + \delta[w])$;

end

if $w \neq s$ then

$C_B[w] \leftarrow C_B[w] + \delta[w]$;

end

end

end

**Algorithm 3:** Parallel Exact Betweenness Centrality
8 Future Work

This section details future work that can be explored in relation to the data sets. In no way are the calculations and analysis of the data sets complete. Each of the future work items is grouped into related topics.

Adjacency Matrices

It is well-known that if \( G = (V, E) \) is a graph, and \( M \) is its adjacency matrix representation, \( M^k_{i,j} \) for \( k > 1 \) is the number of paths of length \( k \) between vertices \( i \) and \( j \). One idea for future work is to investigate the average number of \( k \)-paths for each \( i \in \{2, 3, \ldots\} \). Is there a particular pattern for increasing values of \( i \)? This is useful and relevant to the discussion on characteristic path length.

According to [19], if matrix multiplication is solved in \( O(n^\omega) \) time, then their algorithm solves the APSP problem in \( O(n^\omega \log(n)) \) time. The current best value for \( \omega \) is 2.3728 by [5], but is not practical. Is it possible to decrease matrix multiplication time based on the structure of the graphs? A speedup would allow for Blondel et al.’s algorithm for similarity to be more tractable. For reducing computation time, one can look into Strassen’s algorithm for matrix multiplication, which runs in \( O(n^{\log_27 + o(1)}) \) time. For example, if a graph had 60000 vertices, the running time would decrease by a factor of about 8.

Characteristic Path Length

One can look at a more fine-grained approach to viewing the CPL values for a subset of a given year; for example, create a correlation graph over each set of 4 weeks (i.e. each month) of the year. over a given year. Is there a pattern over time with regard to the values? Possibly compare specific months of different years, or employ some of the patterns seen with induced time lag (see Section 5).

We used a “randomized” method for cosine similarity - for reducing running time one can use the same process for the CPL algorithm. The idea is to sample \( k \) random pairs of vertices, and find the shortest path length between them. Single source shortest path for undirected graphs is solvable in linear time [21], so this algorithm should only take \( O(k|V|) \) time instead of the exact \( O(|V|^3) \) described in Section 6. How much of a performance improvement can be made to this algorithm? One can note that even if all \( (|V|^2) \) pairs of vertices are chosen, the algorithm would have the same asymptotic running time, but not have to hold the entire adjacency matrix simultaneously in memory.

Centrality

There are numerous algorithms for parallelizing betweenness centrality, such as the one presented in [15]. One can use this algorithm and compare running times with the serial version (as provided in the code module) of the algorithm. However, this parallelized algorithm may be slow because it is designed for any general graph. Is it possible to use the structure of these graphs to improve running time? The same can be said for closeness centrality, as well as an approximation algorithm.

After the work in [16], showing that NSIDC graphs are small-world, we also classified the NSIDC graphs as scale-free. However, there does not seem to have been any work that makes a connection between the centrality properties exhibited by these graphs and that of scale-free graphs - one can
investigate if there is such a connection. There most likely is because not only do these graphs have strong indicators of these properties, but also the properties themselves are defined in a mathematical manner.

**Connected Components**

As we have said before, there is one connected component with most of the vertices in each of the graphs. An interesting question is to see why this is so. In regards to this very large component, any non-directly connected vertices are not highly correlated, but any directly connected ones are (by definition). One can look at distributions of correlations for each vertex and notice if there are any patterns, as was done with degree distributions.

**Contours and Convex Hull**

A code module, ConvexHull.java, was created for this section. The convex hull of a set of points is defined to be the smallest convex set that contains that set of points. The code module, for a given set of points, computes the convex hull, as well as the area of that convex set. In the top level directory, there is a file named convex_hull that lists the year, the number of vertices in that graph, the area of the convex hull of that graph, as well as the \{maximal, minimal\} x and y coordinates of the convex hull point set. As one can see, neither the hull area nor the x and y coordinates change over time, if at all. Therefore, it is preferable to study convex hull on a “smaller scale.

Work has been done to look at whether the NSIDC graphs resemble contour maps. The approach was to look at them in terms of the graph’s structure, and see how well the contours translate to a physical contour. We looked at the vertex with highest closeness value, and computed each layer \(i\) as all of the vertices strictly \(i\) edges away from this vertex. This defines a partitioning of vertices according to their distance from this vertex. One can compute the convex hull area for each of the layers, as well as the maximal and minimal x and y coordinates for all vertices in each layer.

**Graphs**

We only looked at \(r_{\text{threshold}} = 0.9\) in this thesis - what happens if we look at \(r_{\text{threshold}} \in \{0.5, 0.7\}\). One would have to re-compute all of the results given in the thesis (CPL, betweenness and closeness, etc.), but a comparison to the results for 0.9 may give some interesting results.

We described the similarity metric described by Blondel, et. al. [8][7]. The difficulty with implementing this algorithm is not only memory management with multiple adjacency matrices that are gigabytes in size each, but also for matrix multiplication; a simple implementation takes \(O(n^3)\) time, which is equivalent in asymptotic running time to the Floyd-Warshall algorithm. However, even that algorithm is not very tractable with graphs of this size. If one were able to implement this algorithm, one can compare the results with those of cosine similarity, and see which of the two metrics is more applicable to the NSIDC graphs.

A relatively recent graph generation technique for climate data is in [11]. However, the running time in their generation of graphs far exceeds that of the graphs in this thesis (3000 vertices for 4.5 days, whereas this thesis has nearly 60000 vertices for 2 days). Their method also takes into account the physical spacing between vertices, a factor that was not analyzed in depth in this thesis.
Future Work

One could use their results and see if their approach is more applicable to the graphs than using Pearson’s metric.

Vertices

We looked at vertex degrees and the distributions that they have - are there patterns over time? One can also look at the \{maximal, minimal, average\} vertex degree for each graph.

We touched on the physical location of vertices in terms of where the vertex with highest closeness value moves in time - what about other vertices? Also, what can be said about the physical angles between vertices? This question may be useful for considering k-cliques, where \(k \in \{3, 4, \ldots\}\) (a clique is a sub-graph where all vertices in the clique are connected by one edge to all other vertices in the clique).

One metric that has been used recently in applications of graphs is \(r\)-dominating sets. An \(r\)-dominating set is defined for a graph \(G = (V, E)\) as the smallest set of vertices \(R \subset V\) such that \(\forall v \in R, \) every other vertex in the graph is within \(r\) edges of \(v\). One can answer the question of what patterns can one see for different values of \(r\). For “Contours and Convex Hull” above, we gave a definition of layers, so it may be more worthwhile for one to investigate \(r\)-dominating sets from some metric other than the vertex with highest closeness value.


9 Conclusion

The results as given in this thesis were both theoretical and computational advancements in the study of the sea ice data. New techniques were proposed for overcoming technological challenges in previous work.

In Section 2 we first defined connected components and closeness centrality. We then gave an algorithm for finding the vertex with highest closeness value. We made a connection with the coordinate of the vertex with highest closeness value with a well-studied area of ice by [16]. We investigated vertex degrees, and how the graphs resemble scale-free networks (a subset of small-world graphs) due to their power law distribution of the degrees.

In Section 3 we looked at the intersections of the largest connected components of the graphs in various ways. We investigated pairwise intersections for all pairs of graphs, and found that the large majority of vertices stay within both graphs. Next, we measured how many vertices are within a global intersection of i graphs, by counting the number of graphs a particular vertex is within the largest connected component of each graph. We found that nearly half of all vertices are within this connected component of all graphs, and made connections of these results to a physical map.

In Section 4 we looked at a definition of betweenness centrality, and how it is important to information within a graph or network. From running the algorithm for determining betweenness centrality for all vertices, we developed images giving a visual representation of the betweenness values. Since the distribution of betweenness values follows an exponential curve, they follow other work that have seen the same pattern for small-world networks [6], furthering prior research [16] on the NSIDC graphs’ similarity to this model.

In Section 5 we investigated two measures of graph similarity: Blondel et al.’s similarity measure, and cosine similarity. Since the former was largely intractable for the NSIDC graphs, we investigated the properties of the latter. We were able to relate the Watts-Strogatz graph model to the NSIDC graphs in this way, since they have similar average cosine similarity values. In the results, we first showed that the average cosine similarity value given by comparing each graph with the next year exhibits (in nearly all cases) a larger value than the predicted value by the Watts-Strogatz model. We then built 4 graphs, composed of 4 weeks each, and had them be offset by a number of weeks. We concluded that there is a higher cosine similarity correlation (maximal and average) when the gap between the compared years and weeks is larger.

In Section 6 we gave numerous theorems about the characteristic path length of undirected, unweighted graphs, such as upper and lower bounds, and separability of connected components for more tractable memory calculations (with proven correctness). We used the calculated upper bounds with the actual values for CPL to establish a downward trend not only in the actual CPL value of the graphs, but also in how large the actual CPL value compares to this upper bound.

In Section 7 we proposed a parallelized exact betweenness centrality algorithm (with proven correctness) that uses the separability of connected components as described above to its advantage. We also gave results on the improved algorithms as described in Sections 4 and 6.
10 Acknowledgments

I would like to thank Dr. Syrotiuk and Dr. Colbourn for supervising my work and giving advice. They have been extraordinarily helpful through correspondence and always encouraged me to go one step further, to go beyond what has been done before. I thank them for their insight in all of the sections, especially about pursuing the results related to vertex intersections.

This thesis would not have been even remotely possible without all of the help of Wayne Chan, who not only worked in projects related to the creation of the data sets used in this thesis, but also gave me invaluable information through email during my work.

Also, my friends and family have always been supporting of the many hours spent working on this thesis. I appreciate all of the questions that they have asked me and their interest in the work, and I will always be grateful. They have also been very helpful in allowing me to “think back” and to keep the practical focus of the work in mind, about “how do the high-level algorithms and ideas relate back to the original meaning?”
Appendices

A Definitions

Here we give definitions of items used in this thesis.

- **Betweenness Centrality**: a measure of a particular node’s centrality in a graph or network. Specifically, for a node \( v \), the betweenness centrality is defined to be:

\[
\text{BetweennessCentrality}(v) = \sum_{s \neq v \neq t} \frac{\sigma_{st}(v)}{\sigma_{st}}
\]

where \( \sigma_{st} \) is the number of shortest paths from \( s \) to \( t \), and \( \sigma_{st}(v) \) are those that pass through \( v \). Let the quantity \( \frac{\sigma_{st}(v)}{\sigma_{st}} = 0 \) if \( s \) and \( t \) do not have a path through \( v \), or are not all in the same connected component.

- **Characteristic Path Length**\(^3\) (CPL): The characteristic path length of a graph \( G \) is the average of the length of the shortest path between any pair of vertices of \( G \). Specifically, for a graph \( G = (V,E) \), the CPL of \( G \) is:

\[
\text{CPL}_G = \frac{\sum_{i \neq j} d(v_i,v_j)}{|V| \times (|V| - 1)}
\]

for \( v_i, v_j \in V, d(v_i,v_j) \) is the shortest path connecting \( v_i, v_j \) (not included in calculation if no path exists). The numerator of \( \text{CPL}_G \) is sometimes called the Weiner index.

- **Closeness Centrality**: a measure of how “close” a particular vertex is to all other vertices in the same component. Formally, closeness is defined for a vertex \( v \in V_i \), where \( V_i \) is a connected component of an undirected graph \( G \) as:

\[
\text{Closeness}(v) = \sum_{j \neq v, j \in V_i} \frac{1}{d(v,j)}
\]

where \( d(v,j) \) is the length of the shortest path between \( v \) and \( j \).

- **Connected Component**: A connected component, \( CC = (V_{CC}, E_{CC}) \) is a subgraph of \( G = (V_G, E_G) \) such that:

1. \( V_{CC} \subseteq V_G \) and \( E_{CC} \subseteq E_G \)
2. \( \forall v \in V_{CC}, \exists v' \in V_{CC} \setminus \{v\} \) with some \( e \in E_{CC} \) connecting \( v \) and \( v' \).

Since the graphs in this thesis are always undirected, the connected component a vertex corresponds to is always unique. Also, if the degree of a vertex is 0 (an isolated vertex), it is not considered to belong to a connected component.

- **Correlation Graph**: A correlation graph \( G = (V,E) \) is constructed as follows:

  - Add every oceanic, non-land, non-missing data point to \( V \).

\(^3\)Also called “average path length” in the literature.
Definitions

– For ∀(u, v) ∈ V × V with u ≠ v, compute the correlation between the time series for u and v, using Pearson’s product-moment correlation coefficient, defined to be:

\[ r = \frac{\sum_{i=1}^{n}(X_i - \overline{X})(Y_i - \overline{Y})}{\sqrt{\sum_{i=1}^{n}(X_i - \overline{X})^2} \sqrt{\sum_{i=1}^{n}(Y_i - \overline{Y})^2}} \]

– Given is a threshold \( r_{\text{threshold}} \). If \( r > r_{\text{threshold}} \), create an undirected edge between u and v.

(From [10])

• **Degree Centrality**: for a node v, the degree centrality of v is how many edges are connected to v.

• **Dense Graph**: A graph \( G = (V, E) \) is dense if \(|E| \in O(|V|^2)\); informally, the number of edges quadratically (“squared”) corresponds to the number of vertices in the graph.

• **Graph Clustering Coefficient**: A graph clustering coefficient \( \gamma_G \) of a graph \( G = (V, E) \) is the average clustering coefficient \( \gamma_v \) of ∀v ∈ V.

• **Graph Degree**: The degree of a graph \( G = (V, E) \) is the average degree of the vertex degree of v for ∀v ∈ V. The degree of a vertex v ∈ V is the number of incident edges to v, since the graphs are undirected.

• **Sparse Graph**: A graph \( G = (V, E) \) is sparse if \(|E| \in \tilde{O}(|V|)\); informally, the number of edges linearly corresponds to the number of vertices in the graph.

• **Vertex Clustering Coefficient**: A vertex clustering coefficient \( \gamma_v \) of a vertex v in a graph \( G = (V, E) \) is the edge density with all vertices within one edge of v. Formally, the clustering coefficient of a vertex \( v \in V \) is:

\[ \gamma_v = \frac{|\{e_{jk} : v_j, v_k \in N_i, e_{jk} \in E\}|}{|N(v_i)|(|N(v_i)| - 1)} \]

where \( N(v_i) \) is the set of neighbors of \( v_i \).

• **Wiener index**: The numerator of the characteristic path length of a graph \( G \), namely:

\[ D_G = \sum_{i \neq j} d(v_i, v_j) \]

\(^4\)This is an alternative definition given by Watts and Strogatz.
B  Code Module Documentation

We now give documentation on each of the core code modules that provided the results seen in this thesis.

- **BetweennessCentrality.java** - method **allValues** (and **allValuesMaxSet** for the maximal size connected component) takes in a Graph object and returns a HashMap\langle Vertex, Double\rangle object, where each vertex is associated with its betweenness value. The algorithm, on a high level, works on each connected component, and from each of those computes a breadth first search and uses Brandes’ algorithm [9].

- **BetweennessCentralityMax.java** - prints out the year and the largest betweenness value that is located in a file **output\langle year\rangle.txt**, which was created inside the BetweennessCentralityToImage module.

- **BetweennessCentralityToImage.java** - creates two files: **output\langle year\rangle.txt**, and **\langle year\rangle.jpg**. The former is a sorted (in increasing order) list of betweenness values of vertices, and the latter is an RGB image that for each pixel \((i, j)\) corresponds to a linear ordering between value \((0, 0, 0)\) and \((255, 255, 255)\), as was documented in that section.

- **ClosenessCentrality.java** - method **allValues** takes in a Graph object and returns a HashMap\langle Vertex, Double\rangle object, where each vertex is associated with its closeness value. The algorithm, on a high level, works on each connected component, and from each of those computes a breadth first search.

- **ClosenessCentralityContourLines.java** - does one of the following (based on the value of the value **READ**):
  - Reads through the given graph file years, finds the largest connected component of them, finds the vertex in this connected component with the largest closeness value, and returns a HashMap\langle Integer, Set\langle Vertex\rangle\rangle, where each key corresponds to a set of vertices that compromises each of the “levels”. More formally, the vertices with a key of \(i\) correspond to the vertices exactly \(i\) edges from the vertex with highest closeness value. The object is then saved as: **\langle year\rangleDiffLayers.closeness**.
  - Reads through the given years for **\langle year\rangleDiffLayers.closeness** and displays the results.

- **ComputeCharacteristicPathLength.java** - uses the algorithm described in Section 6 to calculate the CPL of an inputted Graph object.

- **ConvexHull.java** - computes the convex hull of an inputted ArrayList\langle Vertex\rangle of a graph.

- **CosineSimilarity.java** - computes and outputs the cosine similarity between two Graph objects with their years as input.

- **DegreeDistribution.java** - creates a .csv file for which each entry corresponds to the degree of a vertex, and the frequency of that degree in the inputted Graph object.

- **GenerateGraphBuildingFiles.java** - creates .txt files that automatically call GraphBuilding to construct the graphs. It was easier to do this process than to manually input numbers.
• **MaximalCPLGenerator.java** - computes and outputs the maximal CPL for an inputted **Graph** object, as described in Theorem 6.2.

• **ReadGraphFile.java** - reads a **Graph** object that was written to disk at the end of the **GraphBuilding** module. Most other code modules that import **Graph** objects use this module. The motivation for writing the graphs to files was to drastically reduce the amount of time spent doing computations on the graphs (without having to re-build every single time).

• **VertexIntersection.java** - reads any number of graphs and computes the number of vertices (as well as which ones) are within the largest connected components of all of those graphs. There is also code within **VertexIntersection.java** that does all pair-wise intersections between all of the graphs; in this case, we do not output the vertices themselves, as most vertices between any pair of graphs are within their largest connected component.

• **VertexIntersectionParser.java** - uses the .txt files outputted by the global intersections in **VertexIntersection.java** and creates images where the corresponding pixel becomes white if it is in the file currently being read.
References


