Master Project Report

Approximation of Betweenness Centrality

Daniel Autenrieth
Computational Social Systems
RWTH Aachen
Aachen, Germany
daniel.autenrieth@rwth-aachen.de

Abstract

The betweenness centrality metric has become an important factor in a wide variety of network analysis tasks. However, since the calculation takes O(|V ||E|) time, it is usually not reasonable for very large graphs to calculate these values. To solve this problem, the values are approximated by different methods. But because these methods introduce error rates into the values, the objective of this paper is to investigate the effect of network properties on the prediction quality. The clustering coefficient, the degree of the nodes, the betweenness centrality itself, and the diameter of the graphs were considered. Different error distributions as well as explainable behavior patterns based on the underlying algorithms could be found. However, no systematic mispredictions could be identified. Therefore, an improvement of the method was not possible. The results were supported by the use of various regression models.

CCS CONCEPTS

• Networks → Network properties • Theory of computation → Graph algorithms analysis; Approximation algorithms analysis

KEYWORDS

Network properties, Betweenness Centrality, Approximation, Error Distribution, Social Networks

1 Introduction

The betweenness centrality metric has become an important factor in network analysis. Whether it is optimizing telecommunication networks or identifying key individuals and potential leaders of terrorist groups, in all these tasks' researchers rely on betweenness centrality, among other metrics, to understand the graphs. A disadvantage of this metric, which is based on shortest paths, is the high computational cost. One of the best known and most widely used algorithms for the parallel calculation of betweenness centrality is the Brandes algorithm [1]. But even this algorithm has a time complexity of O(|V ||E|) and O(|V ||E| + |V |^2 log|V |) for unweighted and weighted graphs, where the input graph G has |V| vertices and |E| edges. Even a relatively small real graph with an assumed number of 4 million nodes and 34 million edges, calculated on a typical workstation, will one whole year [2]. Since this high computing capacity is rarely available, other methods have been developed to reduce the required computing times. Most methods therefore only include a subset of the node pairs in their computations [3–7] or approximate the shortest path computation itself [8, 9].

All scripts and visualizations necessary for the replicability of the project can be found under:
https://git-ce.rwth-aachen.de/daniel.autenrieth97/my-master-project

Research Goal. However, since these methods induce error rates in the data, it was the task of this project to search for systematic errors that are based on the network properties, that lead to misclassification of betweenness centrality values. For this purpose, the following properties were selected for investigation: (1) the clustering coefficient, (2) the degree of the nodes, (3) the diameter of the graph and (4) the betweenness centrality values itself.

Datasets and Methods. Calculated exact betweenness centrality values from [2], which can be accessed here [10], were used as ground truth. Approximated values of three different algorithms: here called Arba, DIAM and RAND2, which will be explained in more detail later, were generated and by comparing them with the exact values, the error rates could be determined.

Approach. The basic approach was first to create visualizations of the error values and a specific property. Trends and anomalies in the plots as a starting point for further investigations. However, as no clear patterns were visible in most of the plots, a final check was carried out using various regression models. In the visualizations with the betweenness centrality values themselves, a deeper analysis was carried out. More details can be found in the corresponding chapter.

Findings. No direct correlation between the network properties and the error distribution could be found. When visualizing the betwenness centrality values, interesting structures were found, which can be explained by the behavior of the algorithm. Overall, this work tried to find correlations especially with respect to the relative error. Since relatively large error rates occur with small

values, an improvement of the sum of the error would not necessarily have been beneficial and was therefore not pursued as a main goal.

2 Methods

First of all, a unified definition of Betweenness Centrality (BC) should be given, on which the following methods are based.

$$BC_G(v) = \sum_{\substack{s,t \in V \\ s \neq t \neq v}} \frac{\sigma_{st}(v)}{\sigma_{st}} \quad (1)$$

Where V is a set of nodes, σ_{st} is the number of shortest (s,t)-pathsand $\sigma_{st}(v)$ is number of those paths passing through some node v other than s,t. Or Betweenness centrality of a node v is the sum of the fraction of all-pairs shortest paths that pass-through v.

2.1 Brandes

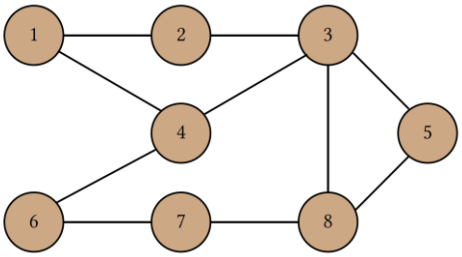


Figure 1: Example graph [2]

To understand more precisely where the problem of the quadratic computing time originates and what many of the approximation methods refer to, a short insight into Brandes' algorithm will be given in the following.
The algorithm can be divided into two main phases. in the first one a breadth-first search is performed, and the number of all possible paths is calculated. In the second phase, reverse breadth-first search and the following formula is used to calculate the pair dependency:

$$\delta_{s\blacksquare}(v) = \sum_{v \in P_s(w)} \frac{\sigma_{sv}}{\sigma_{sw}} \cdot (1 + \delta_{s\blacksquare}(w)) \quad (2)$$

Where $P_s(t)$ is the parent node. To make this a little clearer, here is the illustration of the results of the steps.

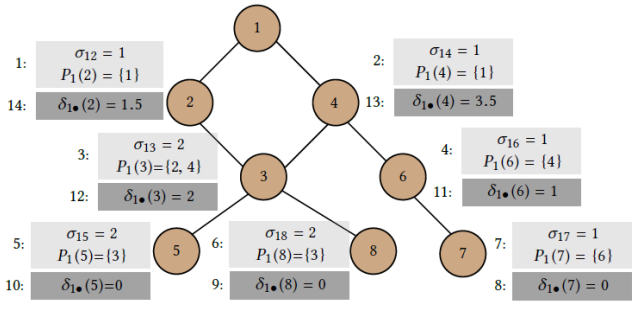


Figure 2: Compute $\delta_{1\blacksquare}(v)$ for example graph [2]

In this example, the values for a run starting from node number 1 are shown. To get the actual betweenness centrality value, the same process must be performed starting from all other nodes. This means that a number of nodes traversals are required to produce the exact results which then leads to the time complexity of O(number of nodes · time of traversal).

2.2 Approximation methods

In principle, the methods used can be divided into two categories, based on the basic operation of the approximation. First, there is the so-called "source sampling". A subset of nodes is randomly selected as starting points. The RAND2 algorithm [5] belongs to this category. the algorithm is based on a paper by Geisberger and Sanders [5]. The algorithm is based on uniform random sampling with lower contributions for near to pivots nodes. This means that due to overestimation of the values of nodes that are close to the starting point, a correction factor has been added to reduce this error. This can be represented by the extension of the Brandes algorithm:

$$\delta_s(v) = \sum_{v \in succ(w)} \frac{\mu(s,v)}{\mu(s,w)} \frac{\sigma_{sv}}{\sigma_{sw}} \cdot (1 + \delta_{s\blacksquare}(w)) \quad (3)$$

Where $\mu(s,v)$ ist the shortest path distance and $\frac{1}{\mu(s,w)}$ is the correction factor introduced by Sanders.
However, node pairs can also be sampled instead of individual nodes. This is called node-pair sampling and both the Abra and DIAM algorithms belong to this category.
The DIAM algorithm is based on a paper by Riondato and Kornaropoulos. In their paper they show that with a probability of at least $1-\delta$ the additive error is at most ε. both δ and ε are parameters that are set and influence the sample size and therefore the computation time. The sample size is also based on the vertex-diameter (VD), which is the size of the shortest path in G with maximum size. In graphs with unitary wiegths, the vertex diameter is equal to number of edges composing the longest path plus one. The whole equation for the sample size r, shows the direct connections between the parameters:

$$r = \frac{(1-\delta)}{\varepsilon^2}([log_2(VD(G) - 2)] + 1 + ln\frac{1}{\delta}) \quad (4)$$

The name chosen here, derived from the BeBeCa paper [2], was adopted because this algorithm is dependent on the diameter of the input graph. In the coding phase an implementation of networkit [11] was used, which is based on the same concept. The main logic follows those steps: 1. Sample a pair of nodes u, v of distinct vertices uniformly at random. 2. compute set Suv of all shortest paths. 3. select path p from Suv uniformly at random. 4. Increase value of the betweenness estimation of each internal vertex of path p by $1/r$.
The Kadabra algorithm [12], called Abra in the following, also uses node-pair sampling, but enhances this method with the adaptive sampling concept. Adaptive or also progressive sampling means that the number of samples that are required is not static but the depends on the data sampled so far. Therefore the algorithm will sample until a defined stopping condition is met. Also, applicable here: the algorithm will approximate the values with an additive

error of at most ε with a probability of at least $1-\delta$. But in addition, it will only take nearly-liner time to compute with consistent parameters ($O(mn^{1-\epsilon}), \epsilon > 0$). The original implementation of Networkit can be found in [11].
The stopping condition is dependent on the maximum number of samples and per-vertex probability constants. The algorithm can be used to approximate k most central nodes very efficient by using different confidence intervals.

3 Graphs

During the project, the 5 graphs listed below were mainly used. With special emphasis on ca-GrQc, ca-HepPh, ca-HepTh.

Graph	|V|	|E|	AvgDegree	Diameter
Ca-GrQc	4,158	13,428	6.46	17
Ca-HepTh	8,638	24,827	5.75	17
Ca-HepPh	11,204	117,649	21.00	13
email-Enron	33,696	180,811	10.73	11
com-Amazon	334,863	925,872	5.53	44

Table 1: Graphs [2]

ca-GrQc is a collaboration network between authors in arXiv, published under category "General Relativity and Quantum CosmologyCollaboration network of Arxiv High Energy Physics Theory". Ca-HepTh is Collaboration network of Arxiv High Energy Physics Theory [13]. Ca-HepPh is another Collaboration network of Arxiv High Energy Physics from the year 2003 [14]. Email-Enron is famous network that captures email communications before the Enron scandal. Finally, com-Amazon is a customer-product network created by crawling the Amazon online store [2]. All these graphs are undirected and unweighted graphs, which can be downloaded here [10].

4 Network properties

4.1 Clustering Coefficient

n graph theory, a clustering coefficient is a measure of the degree to which nodes in a graph tend to cluster together [15]. In a graph G=(V, E) with a set of V vertices and a set of E edges, the clustering coefficient is defined as:

$$C_i = \frac{2\left|\{e_{jk}: v_j, v_k \in N_i, e_{jk} \in E\}\right|}{k_i(k_i-1)} \quad (5)$$

Where e_{jk} connects the vertices j and k, which are neighbors of i and there exist a maximum of $k_i(k_i-1)$ links among the vertices within the neighborhood. A clustering coefficient of 1 therefore means that the neighborhood is fully connected and a coefficient of 0 means that none of the possible connections between neighboring nodes exists.
As can be seen in figure 3, the number of instances decreases as the clustering coefficient increases except for peaks in at zero and one. this behavior can also be observed in the histogram in figure 4. These observations hold for all graphs discussed here. The images of the ca-GrQc graph shown here serve only as an illustration. Furthermore, the right figure shows the distribution of the clustering coefficient by degrees. As can be seen, the clustering coefficient tends to decrease with increasing degree. However, in the graphs ca-GrQc, ca-hepPh and ca-HepTh we can also observe another line of which is separated from the main component but follows the same direction.

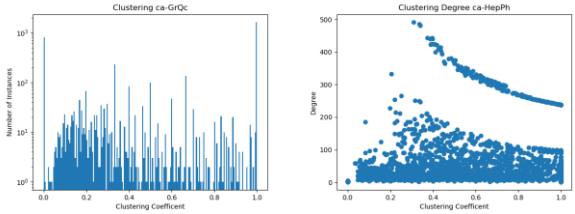


Figure 3: Clustering coefficient vs Degree and Distribution [16]

There are only minor differences in the distribution of the error between the graphs and the methods. The amazon graph is excluded here as well as in all other examples because it often represents a special case.

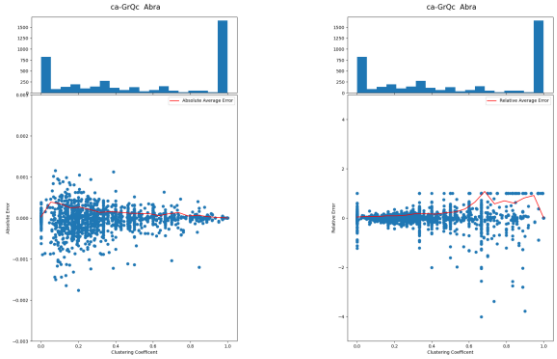


Figure 4: Clustering coefficient and Error [16]

As can be seen in Figure 4, when looking at the absolute error, there is a clear peak in the low range of the clustering coefficient. However, this is not as significant when looking at the right plot with the relative error. This plot indicates that some nodes with a high betweenness centrality value have a low clustering coefficient and distort therefore the picture. Hence, the absolute values are high, but the normalization with the value results in a different interpretation. Consequently, one can say that in general nodes with a higher clustering coefficient have a larger error in relative terms. In the following image you can also see the distribution of the nodes in terms of degree and clustering coefficients. Where the color indicates the relative error. In this representation the nodes which cause the amplitude in the high values are clearly recognizable.

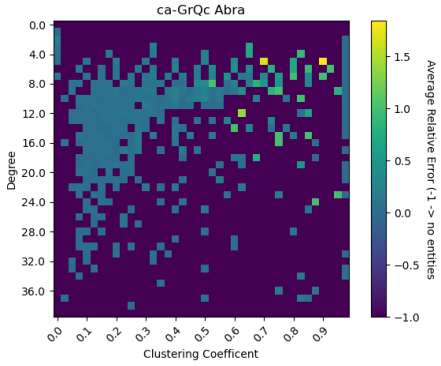


Figure 5: Clustering Coefficient with Degree and rel. Error [16]

More images with the exact distribution of the clustering coefficient and the degree, as well as other representations analyzing the clustering coefficient further can be found, like all other images, in the git repository.

4.2 Degree

The degree of a node indicates the number of connections a node has. A node with two neighbors has a degree of 2 in an *undirected* network.

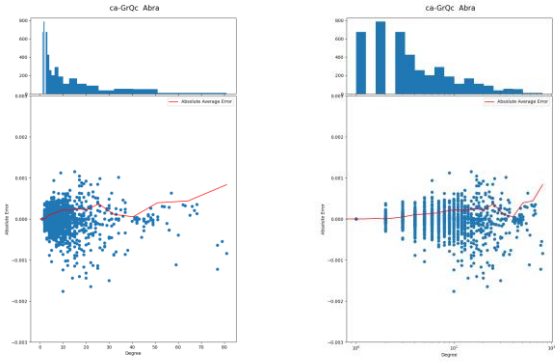


Figure 6: Degree Error Rate (linear/logarithmic scale)

Figure 6 shows the degree against the absolute error. The histogram at the top shows the number of nodes in this range. The figure on the left uses a linear x axis scaling and the figure on the right uses a logarithmic scaling to better examine the range near zero. An overall tendency can be seen. With increasing degree, the error also increases on average.

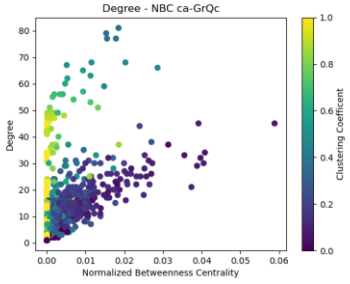


Figure 7: BC vs Degree with clustering coefficient [16]

Looking at the distribution of the nodes with respect to their degree to the normalized betweenness centrality value, trends can be observed here as well. Nodes with a higher degree have a higher BC value on average. Interestingly, some graphs also show that there is a second cluster with nodes with higher degree values. But also, the second cluster follows the trend. This observation applies equally to all graphs analyzed. For this reason, the representation of the ca-GrQc graph is also only an example.
However, the clear tendency that was previously observed in the absolute error is lost when the relative error is considered. The different graphs and methods show different peaks and no generalized trend can be determined.
Since all visualizations should be viewed to confirm this, no single visualization is shown here, but only the folder "Plots\Degree\Relative Error\..." is referenced.

4.3 Configuration Model

In order to isolate the influence of the degree, a configuration model was used in the following and the important parameters were considered again. As seen before, there is a correlation between the degree and other parameters like the clustering coefficient. By using a configurations model, the original degree distribution of the graph can be preserved and at the same time other effects can be suppressed by relinking the connections. Figure 8 shows the effect of the now random network on the clustering coefficient.

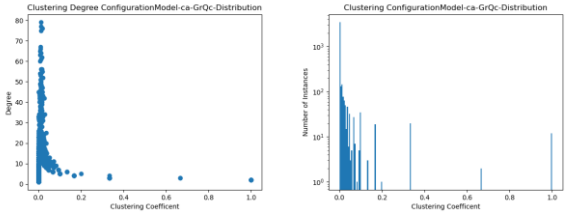


Figure 8: Collapsed Clustering (Degree) Distributions [16]

In figure 9, it can be observed that the insights gained from figure 6 still hold true. Even if other effects are eliminated, a slope with increasing degree and an absolute error remains. For the relative error, there are still strong fluctuations, although a peak can be seen at the low degree nodes.

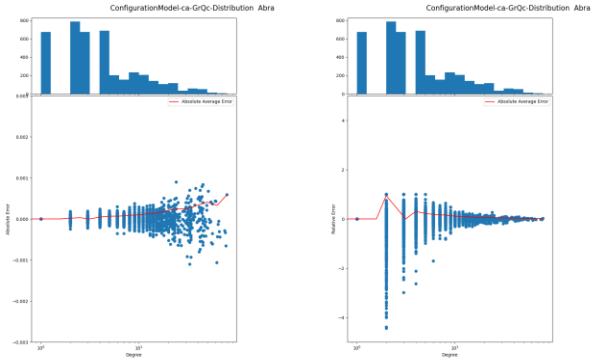


Figure 9: Degree vs (abs./rel.) Error with Configuration model

Also, the observations described in the chapter Betweenness centrality hold after a re-linking of the graph.
Nevertheless, it must be emphasized that even if the structure has not changed significantly, the absolute error has decreased. In the example of the ca-GrQc graph, from 0.314 to 0.235. A significant decrease that indicates the effects of the other parameters. Furthermore, a change in scaling can also be observed in the direct comparison of the different representations.

4.4 Diameter

The diameter as the longest shortest paths is together with the graph size (number of nodes) and important factor to understand the connectivity of a graph. An attempt was made to analyze the effects of the diameter on the accuracy of the approximation methods. For this purpose, different types of graphs were generated. Among others Barabasi Albert and Erdos Renyi graphs.

$$p = \frac{c}{N} \quad (6)$$

Since connectivity c cannot be separated from the number of nodes N and, the individual effects cannot be separated from each other. This test series was postponed. Therefore, no further details are given here. Earlier experiments can be found in the archive folder. However, there are no noteworthy results.

4.5 Betweenness Centrality

An important factor is the error distribution of the betweenness centrality itself.
A pattern which is only visible with a logarithmically scaled axis can be seen in node-pair sampling methods.

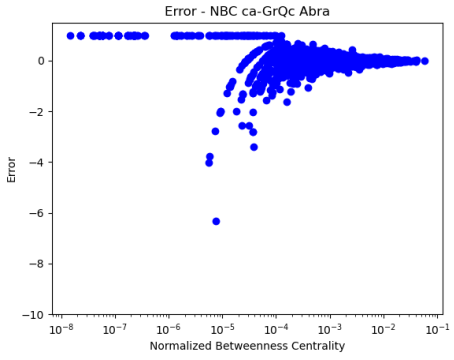


Figure 10: NBC vs relative Error

This shape looks like a function in the format: $y=a/x$.
In order to be able to determine more precisely which function this cluster of points follows and to be able to draw possible conclusions from this, separating lines that were created by example points were first introduced. All nodes lying between the two separation lines were counted into a cluster and could then be considered individually. If one visualizes the process, the following colorful picture emerges.

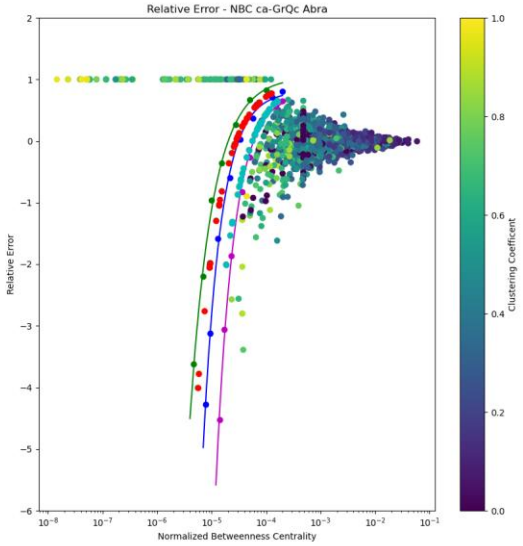


Figure 11: Investigating node clusters [16]

The selected points for the dividing line are shown in green. These points are then approximated by the green line. Together with the dark blue and purple lines and points, they form the dividing lines between the node clusters. The mapping of the individual nodes was visualized with the colors red for the first cluster and light blue for the second cluster. With the now separated points, it could now be proven that the form has the following basic function:

$$rel.Error = \frac{BC-Approx}{BC} \quad (7)$$

The relative error shown is made up of the absolute error = BC – approximated value and is then divided by the BC. To be able to follow the next step, a short look into the Kadabra algorithm is necessary.

Algorithm 1: our algorithm for approximating betweenness centrality.

Input : a graph $G = (V, E)$
Output: for each $v \in V$, an approximation $\tilde{b}(v)$ of $\mathrm{bc}(v)$ such that $\Pr\left(\forall v, |\tilde{b}(v) - \mathrm{bc}(v)| \leq \lambda\right) \geq 1 - \delta$

$\omega \leftarrow \frac{c}{\lambda^2}\left(\lfloor \log_2(\mathrm{VD} - 2) \rfloor + 1 + \log\left(\frac{2}{\delta}\right)\right)$;
$(\delta_L^{(v)}, \delta_U^{(v)}) \leftarrow$ `computeDelta`();
$\tau \leftarrow 0$;
foreach $v \in V$ **do** $\tilde{b}(v) \leftarrow 0$;
while $\tau < \omega$ *and not* `haveToStop` $(\tilde{b}, \delta_L, \delta_U, \omega, \tau)$ **do**
  $\pi =$ `samplePath`();
  foreach $v \in \pi$ **do** $\tilde{b}(v) \leftarrow \tilde{b}(v) + 1$;
  $\tau \leftarrow \tau + 1$;
end
foreach $v \in V$ **do** $\tilde{b}(v) \leftarrow \tilde{b}(v)/\tau$;
return $\tilde{b}$

Pseudo Code 1: Algorithm 1 of the (Kad)Abra [12]

Each node is initialized with zero. If the node is found by a sampled path, its value is incremented by one. If a sufficient number of sampled paths has been reached, all values are normalized by dividing them by the total number of selected paths. Now assuming that the first node cluster represents all nodes that have been found once. With this assumption one can calculate the number of sampled paths. If each node in the cluster returns the same number

of paths, one can be sure that this is the correct solution. Since this is exactly the case and the same works for the second cluster, the form of the error can be explained by the algorithm.
In summary, the shape of the error distribution is related to the number of times a node has been sampled.
Even if this is the explanation, there is unfortunately no direct solution to this problem. As can be seen in Figures 10 and 11, the relative error tends to increase when a node is sampled less frequently. This can be easily understood mathematically but can only be solved by methods that sample the specific nodes more often or in a different way. For this reason, an attempt was made to improve the error rates by including further (previously investigated) parameters (see Regression).

5 Regression

Regression analysis is a set of statistical processes for estimating the relationships between a dependent variable and one or more independent variables [17].
In order to ensure that no patterns were neglected in the visual analysis, correlations are to be checked with the help of various regression models. On the one hand, these models allow insight into the effects of different parameters, and on the other hand, an original approximation can be improved if it depends on the selected factors.
Initially, a simple linear regression model was used, which was then extended by interactions between the individual variables. The models were trained with the data from the three graphs (ca-GrQc, ca-HepPh, ca-HepTh). If there is a fundamental systematic connection between any variables, this should also occur in different runs with different graphs.
The models were trained with the variables from table 1:

	MODEL 1	**MODEL 2**
Features	ApproxValue maxDegree NumberNodes NumberEdges Degree ClusteringCoeff	TimesSampled maxDegree NumberNodes NumberEdges Degree ClusteringCoeff SampledPaths
	MODEL 3	**MODEL 4**
Features	Diameter maxDegree NumberNodes NumberEdges Degree ClusteringCoeff	maxDegree NumberNodes NumberEdges Degree ClusteringCoeff DegreeCentralitiy

Table 2: Models and variables

The first model contains the approximated values of the Abra method, the highest degree of all nodes, the number of nodes, the degree of the specific node and the clustering coefficient of the node. first of all, it must be tried to include the previous parameters such as the clustering coefficient doer the degree. Then it was tried whether the connection between the frequency of being sampled and the total number of sampled paths can be created. In the third model, an attempt was made to obtain the betweenness centrality value only by means of known network properties and in the fourth model a new metric was tested: the Degree Centrality. The degree centrality for a node v is the fraction of nodes it is connected to [18] – so the degree in a normalized form. The different models produced the following results:

	MODEL 1	**MODEL 2**
without Interactions	0.994	0.939
With Interactions	0.994	0.994
	MODEL 3	**MODEL 4**
without Interactions	0.156	0.323
With Interactions	0.479	0.702

Table 3: Models and R^2

The exact results can be found in the “Regression\Saves Different Models”- folder. Due to the limitations, only the most important findings are summarized here. These results can be interpreted as follows. The approximated values have a factor of 0.9844, which is the explanation for almost all changes in the values. Even though several interactions have statistical significance, their values are almost negligible. The linear regression model 2 was able to achieve as good a result as model 1, in which the approximated values were given directly, by using interactions. The attempt to determine the values purely by network parameters did not work. Even if 0.7 would be a step in the right direction, such a method could not be used in real operation. For this reason, we must unfortunately summarize that no new correlations have occurred, and the values have not really improved. In the git repository there is the folder "Plots\Configuration Models" where the other plots were reproduced with the new method. One can see improvement in some lower degree nodes but significant worsening in others. You get the correct value for most nodes with a BC value of 0 - but the general error does not change. Summarized, there are slight deviations but nothing that could be directly called an real improvement.

6 Conclusion

During this project work, different properties such as degree, clustering coefficient, diameter and betweenness centrality were investigated. The main part of the work consisted in the creation of the different plots. Also it was tried to create a kind of library from the accumulated code parts so modules can be reused. Even if the goals set at the beginning of the project have been achieved, there is unfortunately no strong correlation between the individual factors.
To improve the algorithm further, other methods have to be found. There is also the possibility that correlations exist outside the linear range, which can only be detected with high-order functions. By

using extended regression models with significantly more data, these correlations could be found.
However, it is questionable whether the search for such possibly non-existent correlations is a meaningful further step. However, what could be an interesting study is the impact of different graph types on the error distribution. Only briefly mentioned before, however, the amazon graphs are always underestimated by the used methods. One explanation could be for example the extremely large diameter of the graph and the resulting structure.

Difficulties & Learnings

One problem that arose at the beginning of the project was the planned use of the software from the BeBeCa paper. Since the software libraries on which the software is based are no longer up to date and have been translated for the most part in python, it was clear even after several attempts to renew the existing software that another library was needed. Fortunately, many of the required approximation methods are already available in an improved version, which makes a direct comparison with the paper impossible. Also, not all methods used in the paper could be substituted.
One of the biggest challenges during this project was the ever-growing amount of data and folders. Since I hard-coded all naming and references in the beginning, it became more and more difficult to deviate from the existing folder structure or naming conventions. Even after restructuring a large part of the code, there are still functions that I would have liked to make more agile and more functions that I would have liked to combine into one module. in retrospect, a clear naming structure and outsourced static variables would have been a good basis for further development. Even if it is difficult to estimate in which direction the project will run and what will be needed in the future, I would definitely invest more time in the structuring at the beginning next time.

Folder structure

In the following, the folder structure will be briefly explained to simplify the reproducibility. On the first level are all the main python scripts used to create the plot, models, and graphs. There is a standard structure that is repeated for all graphs. This looks like this:

```
.
└── Some Folder/
    ├── Errors
    ├── Approx_Betweenness
    └── Exact_Betweenness
```

This structure can already be seen on the first level. Here the data for the original graphs were stored. But this structure is also repeated for the generated graphs. For example, looking in the folder “Graphs Generated\Erdos Renyi” – folder, one can see the same structure.
In the plots folder you will find all important visualizations that were created during the project. Since the folder structure extends deep down, it is not discussed further here, as it is hopefully an accessible pattern.
In the archive you will find older scripts and data. Which were created during the project but have no further value. They are only valid as proof of work. Especially the attempts concerning the diameter are in this folder.

Software introduction

To be able to execute the most important functions and generate the first representations, the *Pipeline.py* can be used. With this script a basis was created with which the most important functions and modules for the generation of data and visualizations for the original graphs can be executed quickly. I recommend this script as a starting point if one wants to familiarize oneself with the work.
Scripts that are not called directly or indirectly from the pipeline are the following:
CombinedError.py: Simple function to calculate the total Error of a file
ConfigurationModel.py: This module contains various functions that use configuration models, generate data, and visualize them. The main function can be used to execute all these steps in sequence. Settings can be found at the end of the script.
GetNodeBranches.py: This module was used to perform the analyses for the Betweenness Centrality. More detailed descriptions of the process can be found in chapter 4.5. Note: The sampled points used for the bounding functions make the script look large and confusing. These points can be easily ignored.
Other smaller scripts that only perform the function in their name: *GenerateConfigurationModelGraph.py, GenerateDegClustHeatmap.py, GeneratePlotsErrorDiameter.py*. for generated graphs there are variations of scripts as the naming conventions differ.

Naming conventions

<u>Generated Graphs:</u>
Configuration Models:
{type of graphs}_{number of nodes}_{diameter}_{degree}_{run}
Erdos Renyi grahs:
{type of graph}_{number of nodes}_{diameter}_{run}
Or
{type of graph}_{number of nodes}_{diameter}_{average degree}_{run}
Exact-Files:
Exact_{name of used graph}_Abs_norm__{Normalized? [True/False]}
Error-Files:
Error_[graph name}_{method name}

<u>Original graphs:</u>
Exact-Files:
{graph name}
Error-Files:
Error_{graph name}_{method name}
Approximation-Files:
Approx_{graph}_{method}_norm_{Normalized? [True/False]}

ACKNOWLEDGMENTS

This work was realized with the support of my supervisor Felix Stamm. I would like to thank him very much for his time, his competence and his friendly advice.

References

[1] Ulrik Brandes, "A faster algorithm for betweenness centrality," in *Journal of mathematical sociology 2001*, pp. 163–177.

[2] Ziyad AlGhamdi, Fuad Jamour, Spiros Skiadopoulos, and Panos Kalnis, "A Benchmark for Betweenness Centrality Approximation Algorithms on Large Graphs," *SSDBM '17, Chicago, IL, USA, June 27-29*, 2017, doi: 10.1145/3085504.3085510.

[3] Matteo Riondato and Eli Upfal, "ABRA: Approximating betweenness centrality in static and dynamic graphs with Rademacher averages," 2016.

[4] Matteo Riondato and Evgenios M Kornaropoulos, "Fast approximation of betweenness centrality through sampling," Data Mining and Knowledge Discovery 30, 2 (2016), 438–475.

[5] Robert Geisberger, Peter Sanders, and Dominik Schultes, "Better approximation of betweenness centrality," Society for Industrial and Applied Mathematics, Proceedings of the Meeting on Algorithm Engineering & Expermiments 90–100.

[6] Ulrik Brandes and Christian Pich, "Centrality estimation in large networks.," International Journal of Bifurcation and Chaos 17, 07 (2007), 2303–2318., 2007.

[7] David A Bader, Shiva Kintali, Kamesh Madduri, and Milena Mihail, "Approximating betweenness centrality," International Workshop on Algorithms 124–137, 2007.

[8] Jurgen Pfeffer and Kathleen M Carley, "k-Centralities: local approximations of global measures based on shortest paths," Proceedings of the 21st International Conference on World Wide Web. ACM, 1043–1050.

[9] Martin Everett and Stephen P BorgaŠ i, "Ego network betweenness," Social networks 27, 1 (2005), 31–38, 2005.

[10] Ziyad AlGhamdi, Fuad Jamour, Spiros Skiadopoulos, and Panos Kalnis, *Github BeBeCA -A Benchmark for Betweenness Centrality Approximation Algorithms on Large Graphs.* [Online]. Available: https://github.com/ecrc/BeBeCA

[11] Networkit, *Centrality Methods Overview and Implementation.* [Online]. Available: https://networkit.github.io/dev-docs/notebooks/Centrality.html

[12] N. Borassi, "KADABRA is an ADaptive Algorithm for Betweenness via Random Approximation," 2016.

[13] networkrepository, *Ca-HepTh.* [Online]. Available: https://networkrepository.com/ca-HepTh.php

[14] networkrepository, *ca-HepPh.* [Online]. Available: https://networkrepository.com/ca-HepPh.php

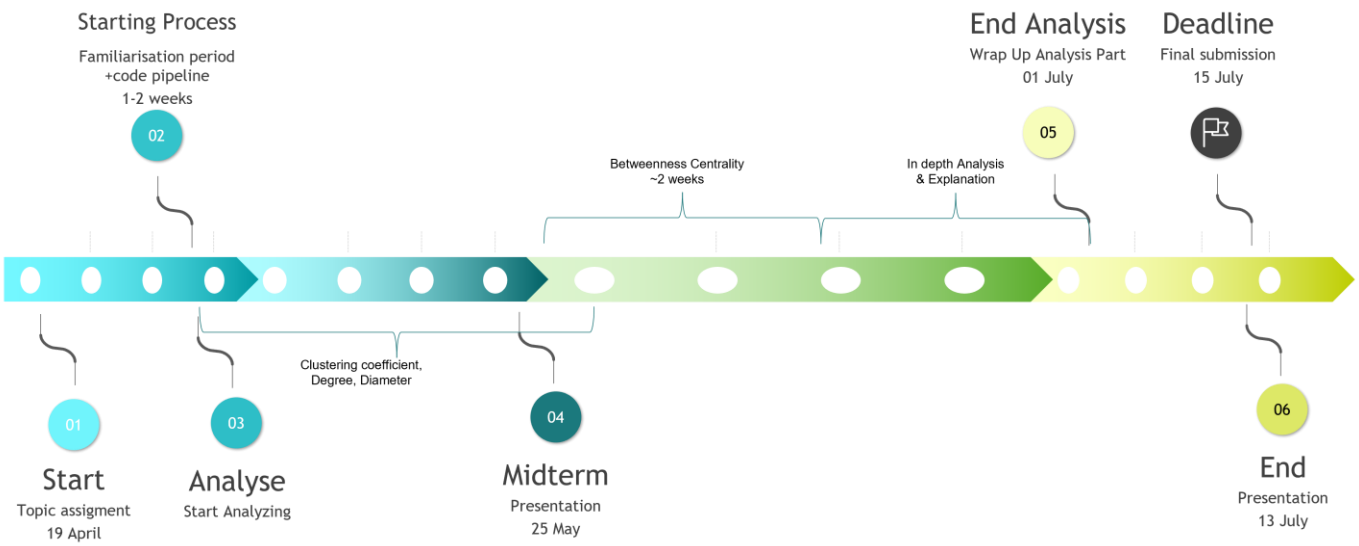
[15] Wikipedia, *Clustering Coefficient.* [Online]. Available: https://en.wikipedia.org/wiki/Clustering_coefficient (accessed: Aug. 23 2022).

[16] Daniel Autenrieth, *Git Repository.* [Online]. Available: https://git-ce.rwth-aachen.de/daniel.autenrieth97/my-master-project

[17] Wikipedia, *Regression analysis.* [Online]. Available: https://en.wikipedia.org/wiki/Regression_analysis

[18] NetworkX, *Degree Centrality.* [Online]. Available: https://networkx.org/documentation/stable/reference/algorithms/generated/networkx.algorithms.centrality.degree_centrality.html

Appendix


Starting Process
Familiarisation period
+code pipeline
1-2 weeks
02
End Analysis
Wrap Up Analysis Part
01 July
05
Deadline
Final submission
15 July
Betweenness Centrality
~2 weeks
In depth Analysis
& Explanation
Clustering coefficient,
Degree, Diameter
01
03
04
06
Start
Topic assigment
19 April
Analyse
Start Analyzing
Midterm
Presentation
25 May
End
Presentation
13 July