UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

FELIPE GRANDO

Methods for the Approximation of Network Centrality Measures

Thesis presented as a partial requirement to obtain a Doctorate degree in Computer Science.

Advisor: Prof. Dr. Luís Lamb

Porto Alegre
2018

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Grando, Felipe

Methods and Tools for the Approximation of Network Centrality Measures / Felipe Grando. – 2018.

114 f.:il.

Advisor: Luís Lamb.

Thesis (Doctorate) – Federal University of Rio Grande do Sul. Postgraduate Program in Computing. Porto Alegre, BR – RS, 2018.

1. Introduction. 2. Complex Network Models. 3. Centrality Measures. 4 Machine Learning. 5. Approximation Techniques for Centrality Measures. 5. Conclusion.

FEDERAL UNIVERSITY OF RIO GRANDE DO SUL
Dean: Prof. Rui Vicente Oppermann
Deputy Dean: Prof. Jane Fraga Tutikian
Postgraduate Provost: Prof. Celso Giannetti Loureiro Chaves
Informatics Institute Director: Profª. Carla Maria Dal Sasso Freitas
PPGC Coordinator: Prof. João Luiz Dihl Comba
Chief Librarian of Informatics Institute: Beatriz Regina Bastos Haro

ABSTRACT

Centrality measures are an important analysis mechanism to uncover vital information about complex networks. However, these metrics have high computational costs that hinder their applications in large real-world networks. I propose and explain the use of artificial neural learning algorithms can render the application of such metrics in networks of arbitrary size. Moreover, I identified the best configuration and methodology for neural learning to optimize its accuracy, besides presenting an easy way to acquire and generate plentiful and meaningful training data via the use of a complex networks model that is adaptable for any application. In addition, I compared my prosed technique based on neural learning with different centrality approximation methods proposed in the literature, consisting of sampling and other artificial learning methodologies, and, I also tested the neural learning model in real case scenarios. I show in my results that the regression model generated by the neural network successfully approximates the metric values and is an effective alternative in real-world applications. The methodology and machine learning model that I propose use only a fraction of computing time with respect to other commonly applied approximation algorithms and is more robust than the other tested machine learning techniques.

Keywords: Centrality Measures. Artificial Neural Networks. Complex Networks Model.

Métodos para a Aproximação de Medidas de Centralidade de Redes

RESUMO

Medidas de centralidades são um mecanismo importante para revelar informações vitais sobre redes complexas. No entanto, essas métricas exigem um alto custo computacional que prejudica a sua aplicação em grandes redes do mundo real. Em nosso estudo propomos e explicamos que através do uso de redes neurais artificiais podemos aplicar essas métricas em redes de tamanho arbitrário. Além disso, identificamos a melhor configuração e metodologia para otimizar a acurácia do aprendizado neural, além de apresentar uma maneira fácil de obter e gerar um número suficiente de dados de treinamento substanciais através do uso de um modelo de redes complexas que é adaptável a qualquer aplicação. Também realizamos um comparativo da técnica proposta com diferentes metodologias de aproximação de centralidade da literatura, incluindo métodos de amostragem e outros algoritmos de aprendizagem, e, testamos o modelo gerado pela rede neural em casos reais. Mostramos com os resultados obtidos em nossos experimentos que o modelo de regressão gerado pela rede neural aproxima com sucesso as métricas é uma alternativa eficiente para aplicações do mundo real. A metodologia e o modelo de aprendizagem de máquina que foi proposto usa apenas uma fração do tempo de computação necessário para os algoritmos de aproximação baseados em amostragem e é mais robusto que as técnicas de aprendizagem de máquina testadas.

Palavras-chave: Medidas de Centralidade. Redes Neurais Artificiais. Modelo de Redes Complexas.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABREVIATIONS AND ACRONYMS

AI	Artificial Intelligence
AMD	Advanced Micro Devices
BFG	BFGS Quasi-Newton
BFGS	Boyden – Fletcher – Goldfarb – Shanno Algorithm
BR	Bayesian Regularization
BTER	Block Two-Level Erdős and Rényi
CB	Betweenness Centrality
CC	Closeness Centrality
CD	Degree Centrality
CE	Eigenvector Centrality
CGB	Conjugate Gradient with Powel/Beale Restarts
CGF	Fletcher-Powell Conjugate Gradient
CGP	Polak-Ribiére Conjugate Gradient
CI	Information Centrality
CPU	Central Processing Unit
CS	Subgraph Centrality
CW	Walk-Betweenness Centrality
CX	Eccentricity Centrality
DDR	Double Data Rate
GD	Gradient Descent
GDM	Gradient Descent with Momentum
GDX	Variable Learning Rate Gradient Descent
IoT	Internet of Things
LCC	Largest Connected Component
LM	Levenberg-Marquardt Algorithm
Mcs	Networks with community Structure
Mer	Simple Random Graphs
Mgr	Geographical Models
Mkg	Kronecker Graphs
MSE	Mean Squared Error
Msf	Scale-Free Networks
Msw	Small-World Model

NN	Artificial Neural Network
Nni	Non-Isomorphic Graphs
OpenMP	Open Multi-Processing Interface
OSS	One-Step Secant
PPI	Protein-Protein Interaction
R^2	Determination Coefficient
RAM	Random Access Memory
RP	Resilient Backpropagation
SCG	Scaled Conjugate Gradient
SGI	Silicon Graphics, Inc.
SSSP	Single Source Shortest Path

SUMARY

1 INTRODUCTION

The increasing development and ubiquity of large-scale networks poses several technological challenges for both researchers and professionals. Network digital and physical connections such as digital links between webpages, friend lists on social networks, wireless vehicular networks, cable connection between routers, streets paths and energy grids raise a growing number of complex questions. As a result, analyzing and understanding network properties is fundamental in several areas of interest and motivates the development and use of several metrics (COSTA et al., 2008; YU et al., 2017). Therefore, the investigation and development of network analysis tools that provide information about the large number of existing networks is clearly relevant and are fundamental tools for many fields, see e.g. (EASLEY and KLEINBERG, 2007; GRANDO et al., 2016; GRANDO et al., 2018).

One class of metrics and some of the most widely used network measurements, given their general applicability, aim at the evaluation, ranking and identification of important vertices by their power, influence, or importance using only the network basic structural properties as input information (FREEMAN, 1978/79). This class of metrics is usually known as “vertex centrality measures”. A collection of metrics composed by several algorithms, each one capturing a different idea of centrality, which are used in typical and fundamental tasks in many computing procedures (COSTA et al., 2008; GRANDO et al., 2016).

Even though its algorithms are polynomial in time, the computation of some of these metrics are computationally expensive becoming a considerable problem when applied to real-world networks that are composed by thousands or even millions and billions of elements and their connections (COHEN et al., 2014). Moreover, there are applications where it is necessary to compute them in real time, where off-the-shelf hardware are used with very limited computing power, or in application environments with dynamical networks that are constantly changing in which is required the computation of such metrics from scratch at each distinct timestamp (BRANDES and PICH, 2007).

These facts rise both a problem and an important research question: “how can we save computational resources to potentiate the application and analysis of networks using centrality measures?”

In order to assist and subserve the use of centrality measures in network applications, I propose and test a methodology and technique to approximate vertex centrality measures using a regression model built by an artificial neural network. Its primal advantage is its capability of

approximating any centrality measures in linear time given the number of vertices of the network by using just a couple of simple input attributes of the analyzed network.

Even though the use of neural networks to approximate the centrality measures is not new in the literature (see e.g. KUMAR et al., 2015), my methodology differs from it in several aspects, starting with different input attributes, neural network structure, algorithms applied and a larger number/size of test case scenarios. Besides, the method developed in my work can be used to approximate any centrality measure and not just a specific one which poses new challenges to find an optimized, yet generalizable, machine learning methodology.

Moreover, I propose a generic methodology which enables anyone to approximate all types of centrality measures, instead of a specific one, using a simple and straightforward artificial neural network model. At the same time, I expand it with a simplified, although scalable, generic and robust, training methodology by using a versatile procedure to obtain the training data with a complex network model called Block Two-Level Erdős and Rényi - BTER (SESHADHRI et al., 2012). With it one can generate networks with diminished size but with the same properties of the huge real networks.

This allows both researchers and practitioners unlimited an easy access to training data for whatever application one may have to tackle. The data obtained with the BTER synthetic networks is proven effective in training the neural network model, which is then able to generalize for the real datasets.

I have tried different configurations and combinations of parameters considering the neural network structure, input information, learning algorithms, meta-parameters, and training data to optimize the artificial neural network performance. Additionally, I use synthetic and real networks with diverse structural properties to englobe as many as we can different test case scenarios. Likewise, I analyzed and compared my approximation methodology with other techniques found in the literature and with the exact centrality values, identifying its drawbacks and strengths.

I show how the machine learning methodology produces competitive results with quality comparable with other approximations methods but – perhaps more importantly - at just a fraction of time and space. The methodology and model can compute the ranking assumed by the centrality measures in linear time/space requirements and under reduced error margins.

Therefore, my research enables the use of complex centrality measures for huge networks by the proposal of a new approximation technique based on a regression model built by an artificial neural learning algorithm while defines a quick setup guide for the overall process to facilitate its use in any application.

The results of my research also include the publication of four papers in important conferences and one publication in a Journal, all of them ranked as A1 by the Capes foundation in the area of Computer Science:

- Two of these papers, entitled: “On Approximating Networks Centrality Measures via Neural Learning Algorithms” (GRANDO and LAMB, 2016) and “Computing Vertex Centrality Measures in Massive Real Networks with a Neural Learning Model” (GRANDO and LAMB, 2018a), were published in the proceedings of the International Joint Conference on Neural Networks (IJCNN);
- One paper entitled: “An Analysis of Centrality Measures for Complex and Social Networks” (GRANDO et al., 2016) in the proceedings of the IEEE Global Communications Conference (GLOBECOM);
- Another one entitled: “On the Effectiveness of the Block Two-Level Erdős-Rényi Generative Network Model” (GRANDO and LAMB, 2018b) in the proceedings of the IEEE Symposium on Computers and Communications (ISCC);
- And the last one, a journal paper entitled: “Machine Learning in Network Centrality Measures: Tutorial and Outlook” (GRANDO et al., 2018) in the ACM Computing Surveys (CSUR).

Several parts of the content of these papers are included in this thesis, including many tables and figures, with small modifications from their presentation in the papers. For a clear reading and to maintain simplicity, most auto-citations were omitted in the thesis.

The first part of this work (Section 2) explains the characteristics of complex models and important studies done in the area. It also presents seven complex network models that will be later used to generate synthetic networks for my experimental analysis and ultimately provided the training data for the neural model.

Section 3 contextualizes, defines and compares the centrality measures, presenting the most relevant metrics and their many research applications. These metrics, specially degree, eigenvector, betweenness and closeness, will be the focus of my research and analysis. They were divided into four groups by their similarity and meaning.

Section 4 gives a brief introduction on machine learning and the most important algorithms. Section 5 explains the organization of my experimental methodology in details, justifying and comparing the choices in each step with other approaches proposed in the pertinent literature. I present, discuss and test the techniques based on sampling methods and

several machine learning algorithms. In this section it also presents the results in details and correspondent analysis of my research.

The last part of this work (Section 6) concludes with the main outcomes and contributions of my work. It also resumes the objectives of this research and outlines possible further investigations with complementary/improvement lines.

2 COMPLEX NETWORK MODELS

Complex networks are ubiquitous in various technological, social and biological domains. Several computer science domains make use of complex and social networks, which can be conceptualized as lying at the intersection between artificial intelligence (AI), graph theory and statistical mechanics, displaying a truly multidisciplinary nature (EASLEY and KLEINBERG, 2010).

The research about complex network has aimed not only to the identification and understanding of network principles, but also to the effective use of their properties and applications. Today, complex network analysis is viewed as fundamental to the understanding of human, social, economic activities, and relationships (LIBEN-NOWELL and KLEINBERG, 2007).

The increasing availability of connected devices, including the unlimited perspectives of the internet of things (IoT), poses several challenges to both research and technology development directly associated to network complexity (ATZORI et al., 2014).

Recent studies on complex networks, which includes computer and social networks, were mainly supported by the availability of high performance computers and large data collections, providing important results and increasing the interest in the area (COSTA et al., 2008).

Research results have shown that technological, biological, and social networks share common properties that cannot be explained by a fully randomized model, such as low diameter ("*small-world*" effect), high clustering coefficients, heavy-tailed degree distribution ("*scale-free*" effect) and presence of community structure (COSTA et al., 2008).

These developments have led to the identification of complex network models capable of stochastically generate networks with similar or related properties. These models were proposed with two main goals (BONATO, 2009):

(i) To understand what underlying effects give rise to such properties;

(ii) To produce synthetic networks with controlled characteristics that may serve as research tools for many disciplines.

Complex network models have been continuously improved to better match and understand the structural properties and features of real-world networks. Such models are useful to generate sample networks with similar characteristics of the real ones that can then be used to improve algorithms and, at the same time, safeguard real data.

Several models have been studied and developed over the last few years aiming at matching features like heavy-tailed degree distributions, low diameter and community structure.

In the next subsections, I briefly summarize the most well-known complex network models, starting by the simpler and elder ones.

2.1 Erdős and Rényi Random Graphs

Erdős and Rényi (1959) introduced the first model methodology that generates simple random graphs. It defined a fixed number of vertices *n* and a probability *p* of connecting each pair of vertices, which also corresponds to the final clustering coefficient of the graph. The higher is the value of *p*, the higher the mean degree and density, and the lower the diameter of the network (ERDŐS AND RÉNYI, 1959).

These simple graphs do not represent real-world networks with high fidelity because they do not present any community structure and because their degree distribution follows a Poisson distribution.

2.2 Watts and Strogatz Small-World Model

In small-world networks, most vertices can be reached within short paths. In addition, these networks show a large number of small cycles, especially of size three.

Watts and Strogatz (1998) proposed the first model to generate networks with the small-world properties. The graph starts with a ring of connected vertices, each one adjacent to its *k* nearest neighbors. Then, with probability *p*, each edge is randomly reassigned to any available position. This relinking method, with an intermediate or small *p* (typically *p* should be lower than 0.5), will create paths among distant vertices while keeping a high clustering coefficient among close neighbors.

The higher is the value of *k*, the higher the vertex mean degree, clustering, and density, although diameter decreases. In addition, the higher is *p*, the lower is the clustering coefficient and diameter (WATTS AND STROGATZ, 1998).

2.3 Barabási and Albert Scale-free Networks

The analyses of large social networks data show that in most of these networks their degree distribution follows a scale-free power-law distribution. Barabási and Albert (1999) explained this property using the fact that networks expand continuously by the addition of new vertices and that these new vertices attach preferentially to vertices already well connected, i.e., vertices with higher degree (or "the rich get richer"). Therefore, they were the first to propose a model that generates network structures with the newly discovered features.

It starts with k number of fully connected vertices and keeps adding new vertices with k connections, defined by a preferential attachment formula. In summary, the preferential attachment formula defines that the probability of a vertex p_i receiving a new connection takes into consideration the degree d of the vertex divided by the sum over the degree of all vertices. This way, high degree vertices have a greater chance of receiving new connections than vertices with lower degree.

The value of k is positive correlated with the mean degree, clustering coefficient, and density, i.e. increasing k also increases the other properties of the network with the exception of the diameter, which is reduced (BARABÁSI AND ALBERT, 1999).

2.4 Networks with Community Structure

Newman and Park (2003) analysis of several social networks showed that such networks are mainly formed by community structures and that the previous complex network models. Moreover, they noticed that each vertex has many connections with vertices inside the same community and few connections with vertices of other, i.e., outside communities. They discovered that, in such networks, high degree vertices tend to be connected with other high degree vertices too. Further, they showed that vertices with small degree are usually connected with vertices with small degree (i.e., they present dissortativity behavior).

They proposed a model that generates random networks which present community structure properties. Their model starts defining c communities and an (uneven) distribution of vertices for each community that represents distinct group sizes. The uneven distribution is important to create the dissortativity behavior (vertices with higher degree tend to be connected with lots of vertices with low degree) seen in the real social networks and also to create communities with different sizes. Each vertex can, and at least some of the vertices must, be

assigned to more than one community. Then, each vertex has a fixed high probability *p* of being connected to every other element of its own communities and zero probability to be connected with vertices to which it does not share a community. Notice that the vertices that were assigned to more than one community are those that link the different communities and are responsible to create a connected graph.

In this network model, the higher the value of *p* and the lower the value of *c*, the higher is the network's mean degree, clustering, and density, although network diameter decreases (NEWMAN AND PARK, 2003).

2.5 Geographical Models

Complex networks are generally considered as lying in an abstract space, where the position of vertices has no definite particular meaning. However, several kinds of networks model physical interactions in which the positions of the vertices characterize a higher probability of interaction with close neighbors than with distant ones. These networks are called geographical networks because in most cases there is an associated cardinal position and coordinates for each vertex because they lie on a geographical space and generally represent physical connections.

Costa et al. (2008) introduced a model for networks with these characteristics that best represent their behavior. In their model the probability of vertices *i* and *j* being connected decays exponentially with distance (usually but not restricted to Euclidian distance) between *i* and *j*.

2.6 Kronecker Graphs

Leskovec et al. (2010) proposed a model in which real world networks are formed by the same substructures, repeatedly and recursively. They also define an algorithm that estimates parameters capable of generating synthetic networks with properties (degree distribution, clustering and characteristic eigenvalues) similar to any given real network. The model starts with an adjacency matrix typically of order two to four. Each cell of the matrix represents the probability that a vertex has a link towards the other in a core substructure.

The larger the matrix the more complex substructures can be exhibited but at the cost of a higher complexity for the algorithm to estimate the proper generation parameters. An

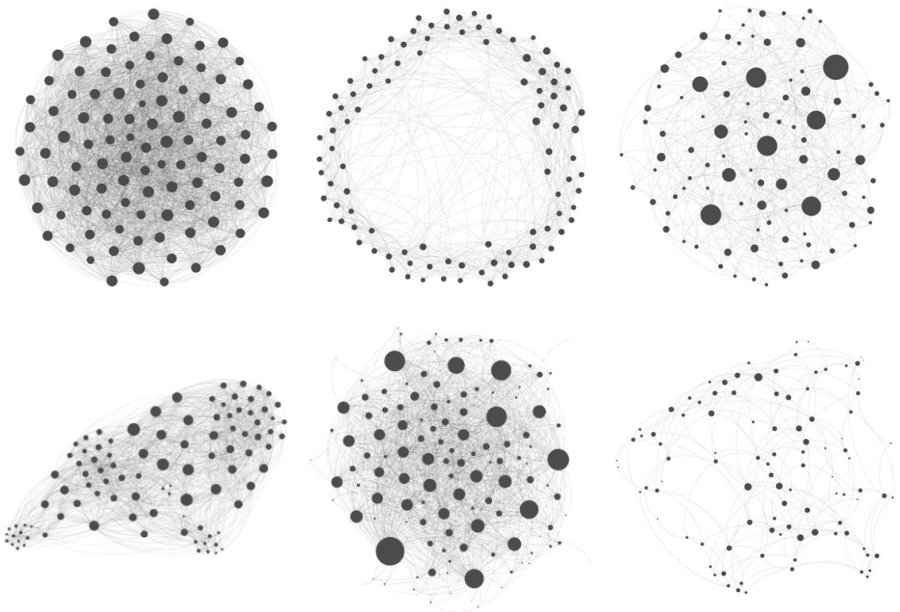
algorithm proposed by the same authors in which a real network is used as basis, estimates these initial parameters by using a mixed strategy based on an exact search and a heuristic to search for the optimal parameters.

Notice that the parameters' matrix is symmetrical for undirected graphs and asymmetrical for directed graphs. To recreate networks of arbitrary size, the matrix is multiplied by itself using the Kronecker product (which gives the name to the model), a generalization of the matrix outer product. The Kronecker product doubles the size of the parameter's matrix and can be repeated until the desired size is achieved, i.e. the matrix's order is close to the number of the desired vertices in the network. The network is then generated by using the parameters (comprised in the cells) in the final matrix, which represents probabilities that each vertex is connected to the other.

This model is one of the more computational costly because it demands the estimation of the initial parameters and also a complex generation algorithm.

I present, in Figure 2.1, six sample networks generated by my implementation of different complex network models. The size of the vertices is proportional to their degree. All networks contain a hundred vertices with the exception of the last one, which contains 128 vertices. The figures were generated with the Gephi software and the vertices were organized by ForceAtlas2 algorithm (JACOMY et al., 2014) in a readable layout.

Figure 2.1 – Synthetic Sample Networks



Simple random graph, small-world model, scale-free model, networks with community structure, geographic model and Kronecker graphs, respectively, from top-left to bottom-right

2.7 Block Two-Level Erdős and Rényi Model

The Block Two-Level Erdős and Rényi (BTER) model (SESHADHRI et al., 2012) generates networks with very similar properties of real networks and can present all the properties presented by the models discussed in the previous subsections. It builds a network based on a degree distribution, which also controls the size of the networks, and, a desired clustering coefficient, which can be uniquely defined for each group of vertices with equal degree, or, globally defined for the entire network. All networks generated by the BTER model present community structures and low diameter (i.e., small-world phenomena).

The BTER model is divided into three steps (KOLDA et al., 2014):

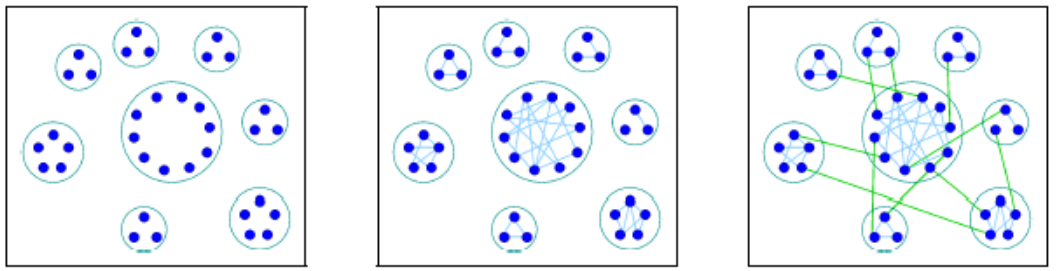
(a) First, the vertices are grouped by degree in communities with size equal to the degree of its members plus one;

(b) Then, each community is considered an Erdős and Rényi graph where the probability of connection among vertices is equal to the desired clustering coefficient;

(c) The last step generates connections proportional to the excess degree of each vertex (number of connections that a vertex needs to match its desired degree). This weighted distribution is based on Chung and Lu graphs (CHUNG and LU, 2002a; 2002b).

Figure 2.2 summarizes these steps.

Figure 2.2 – BTER Generation Steps



(a) Preprocessing: Distribution of nodes into communities

(b) Phase 1: Local links within each community

(c) Phase 2: Global links across communities

Source: SESHADHRI et la. (2015).

BTER is capable to match any real degree distribution and clustering coefficient and it is able to generate huge networks with reasonably low computing time and memory use, employing a highly parallelized algorithm (KOLDA et al., 2014). However, very few

experiments have been carried out to support such a claim. In order to do so, I experimented with several combinations of network parameters to check if BTER is indeed capable of generating synthetic networks with structural properties presented by real networks.

I used several kinds of degree distribution functions, networks sizes and clustering coefficients to analyze the ability of the model to generate networks with the desired properties.

Several investigations have shown that the degree distribution of many real-world networks can be modeled closely by a heavy-tailed distribution or a log-normal distribution (COSTA et al., 2008; EASLEY and KLEINBERG, 2010). The main difference between them is that the log-normal presents a smoother decay than the original heavy-tailed distribution.

I have also selected a linear distribution to further increase the variability of networks properties and thus to check if the model is capable of matching unusual distributions.

Table 2.1 summarizes the formula of each function used to model the degree distribution and the parameter values used in the experiments that were planned to reflect as near as possible real-world behavior. The values generated by each function were considered as proportions (weights) that a given degree appears in a given network size. Therefore, the formula used to define the degree distribution is the same for all networks independently of their size.

TABLE 2.1 – Degree Distribution of Vertices

Distribution	Formula	Parameters
Heavy-tailed	$k^{-\lambda}$	$\lambda = \{1.5, 2, 2.5\}$
Log-normal	$e^{\frac{-(\ln k)^2}{S}}$	$S = \{5, 10, 15\}$
Linear	$ak + \frac{n}{5}$	$a = \{-2-\sqrt{3}, -1, \sqrt{3}-2\}$ n = number of vertices

The second main configurable parameter of the BTER model is the clustering coefficients and for simplicity we choose to use the global clustering coefficient. The global clustering of many real networks belongs to the real number's interval between 0.1 and 0.5, but preliminary tests with the BTER model (and the analysis of the model by its own authors) led to higher choices of clustering coefficients as parameters of the model. This is mainly because the phase (b) of the model decreases the overall clustering of the network defined by the parameter choice. Taking this into consideration, I selected as desired clustering coefficients for the model values belonging to the real number's interval [0.2, 0.7].

There were also other configurable parameters to deal with special cases during the construction of the networks, like degree 1 vertices and groups of vertices with mixed degree.

Those were selected following the suggestions made by the BTER authors. For detailed information about these parameters, please see Kolda et al. (2014).

Moreover, I have added a restriction that all networks generated should be connected. This way metrics like diameter and network analysis metrics such as the centrality measures can be computed normally. The idea of the authors of the model to accomplish connectivity was to remove the smallest components of the graph. However, I preferred to connect all components by generating an edge, uniformly at random, between two vertices of disconnected components. Therefore, I was able to retain the desired network size and solve the problem with a simpler approach.

The final experimental setup contained 2,700 synthetic networks with sizes ranging from a hundred to one thousand vertices. It contained 10 samples of all 27 combinations of parameters selected (kinds of degree distributions and clustering coefficients values).

The size range was chosen to be small because of two main factors:

(i) It is more difficult for a model to reproduce properties in smaller networks, so I tested the model in the hardest problem;

(ii) And, the statistical analysis is costly on larger networks because it will demand more computation time and larger samples to be significant.

Sample instances of networks used in my experiments and analyses are depicted in Figures 2.3 to 2.5. These figures show the networks generated by the BTER model with 100 vertices. The figures were generated with the Gephi software and the vertices were organized by ForceAtlas2 algorithm (JACOMY et al., 2014) in a readable layout. In addition, I set the vertices sizes proportionally to their degree value for an easier visualization of the degree distribution.

Figure 2.3 – Heavy-tailed Distribution with $\lambda = \{1.5, 2, 2.5\}$ respectively from left to right

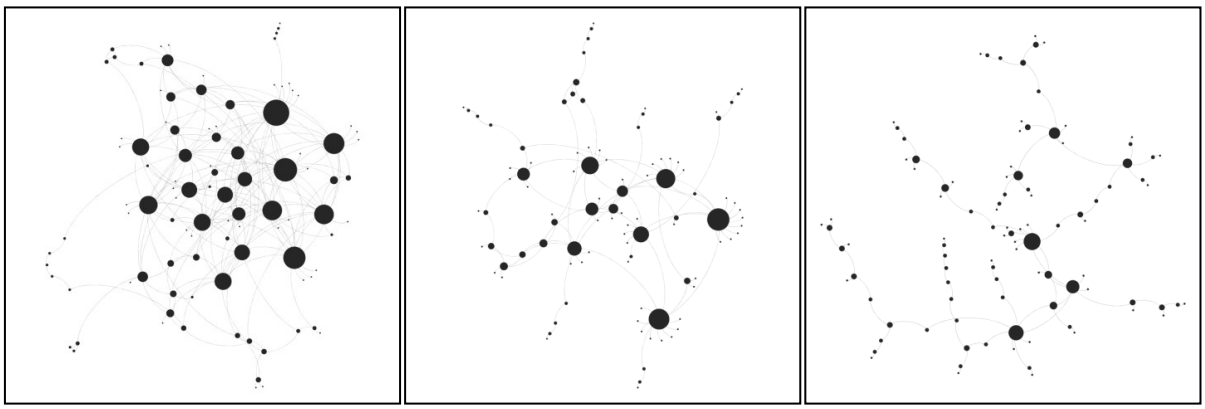


Figure 2.4 – Log-normal distribution with $S = \{5, 10, 15\}$ respectively from left to right

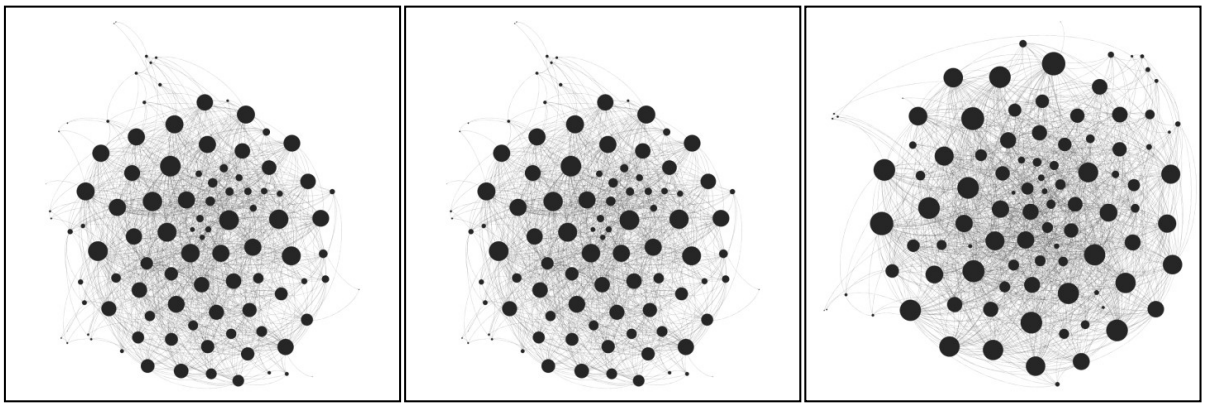
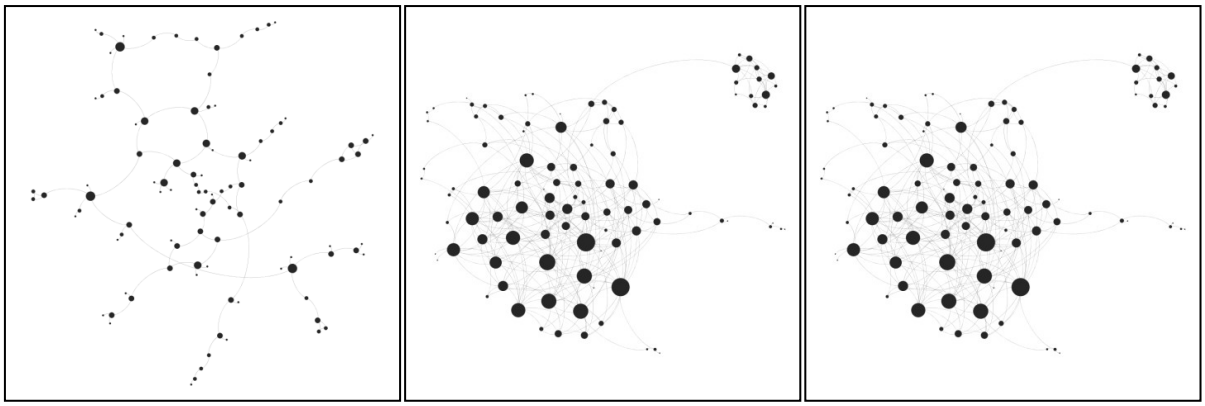


Figure 2.5 – Linear distribution with $a = \{-2-\sqrt{3}, -1, \sqrt{3}-2\}$ respectively from left to right



The communities are difficult to notice in small networks such as the sizes depicted in the Figures, especially for the log-normal degree distributions. But it is difficult also to reproduce visually the graphs of the larger networks in a readable way.

The idea here is to somehow show visually some of the effects of the parameters used in degree distributions for the generated networks. For instance, we can see the correlation between the density of the network and the parameters of each degree distribution (positive correlation for the log-normal and linear formulas and inverse correlation for the heavy-tailed)

Notice also, that the denser is the network the harder it gets to identify the original communities defined in the second step of BTER algorithm because the communities become more and more interconnected and cluttered. Additionally, we can see a clear difference between each kind of degree distribution and between the parameters within a degree distribution in the generated networks. This difference increases even higher with larger sizes of networks.

The main properties of the networks generated by each combination of parameters were computed and are summarized in Table 2.2 and 2.3 as a way to give a more precise idea about

the effect of the parameters and the degree distributions in different sizes of networks. Each cell of the tables represents the mean value over all samples with 100 and 1,000 vertices. The mean variation was insignificant and so was omitted for clarity purposes.

It is important to highlight that the effect of the clustering coefficient (also a parameter of the model) played an insignificant role in the main network properties compared to the degree distribution parameter which means that the clustering is majorly independent from the other properties.

TABLE 2.2 – Properties Summary – Networks with 100 vertices

Distribution	Parameter	Max. Degree	Mean Degree	Diameter	Mean Distance	Density
Heavy-tailed	$\lambda = 1.5$	20.17	4.48	7.63	5.83	0.045
	$\lambda = 2$	13.53	2.43	12.77	9.33	0.025
	$\lambda = 2.5$	9.37	2.04	21.07	16.14	0.021
Log-normal	$S = 5$	35.13	12.93	7.23	5.53	0.131
	$S = 10$	46.70	22.89	4.57	3.45	0.231
	$S = 15$	52.70	27.66	4.17	3.21	0.279
Linear	$a = -2-\sqrt{3}$	5.40	2.22	23.33	17.66	0.022
	$a = -1$	16.80	6.09	9.90	7.38	0.061
	$a = \sqrt{3}-2$	43.03	20.09	4.53	3.50	0.203

TABLE 2.3 – Properties Summary – Networks with 1000 vertices

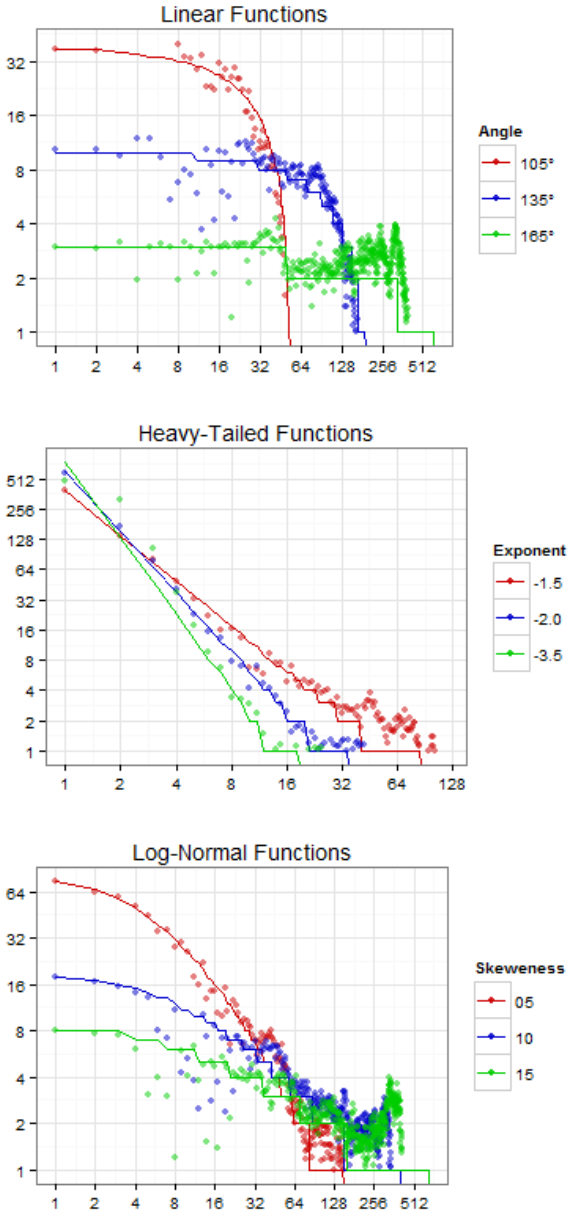
Distribution	Parameter	Max. Degree	Mean Degree	Diameter	Mean Distance	Density
Heavy-tailed	$\lambda = 1.5$	110.37	12.11	10.27	7.24	0.012
	$\lambda = 2$	44.50	3.10	17.13	11.50	0.003
	$\lambda = 2.5$	25.33	2.15	38.07	27.33	0.002
Log-normal	$S = 5$	154.57	28.51	8.40	6.20	0.029
	$S = 10$	349.63	125.27	5.43	3.91	0.125
	$S = 15$	425.43	192.71	4.97	3.59	0.193
Linear	$a = -2-\sqrt{3}$	49.80	16.61	12.33	8.78	0.017
	$a = -1$	172.70	64.54	7.40	5.71	0.065
	$a = \sqrt{3}-2$	401.97	195.96	4.47	3.35	0.196

Despite the fact that the degree distribution formulas did not consider directly the networks size, the properties of the networks are still affected by it as expected. Still, the main features of each kind of degree distribution are present, no matter the size of the network. the effect of the parameters for each degree distribution remains similar in proportion with the size of the network.

It is also clear that the generation of networks by the BTER model with small diameter and low distances is intrinsic to the model process, being independent to the parameter's setup. This fact was already pointed out by the authors of the model (KOLDA et al., 2014).

To check and analyze the accuracy and robustness of the BTER model at following the configured degree distributions I designed the graphs presented by the Figure 2.6. Figure 2.6 presents the desired degree distribution (lines) for a network with 1,000 vertices and the average degree distribution (points) of the synthetic networks generated by the BTER model. The axis scales (quantity of vertices with each degree) of the graphics are logarithmic to assist with the visualization.

Figure 2.6 – Number of Vertices with a Given Degree



We can see that the BTER model was capable of generating networks with a degree distribution close to the one desired for every function that I have used in my experiments. In order to confirm that claim I compute the correlation coefficient between the desired values and the ones realized by the model. The mean Pearson's correlation coefficients with 99% confidence intervals are summarized in Table 2.4.

TABLE 2.4 – Degree Distributions' Correlation

Distribution	Parameter	Correlation
Heavy-tailed	$\lambda = 1.5$	0.996 ± 10^{-5}
	$\lambda = 2$	0.997 ± 10^{-5}
	$\lambda = 2.5$	0.894 ± 10^{-3}
Log-normal	$S = 5$	0.933 ± 10^{-3}
	$S = 10$	0.644 ± 10^{-2}
	$S = 15$	0.435 ± 10^{-2}
Linear	$a = -2-\sqrt{3}$	0.926 ± 10^{-3}
	$a = -1$	0.773 ± 10^{-3}
	$a = \sqrt{3}-2$	0.567 ± 10^{-3}

The correlation values reveal that the denser is the network and higher is the maximum desired degree value, the lower is the performance of the model, i.e., the correlation values were lower. An explanation for that behavior is that the BTER model generates the edges independently from each other. It is inevitable that it will generate lots of repeated edges whenever it needs to generate lots of edges for a high degree vertex. These repeated edges are simply discarded during the process.

It was also interesting to notice that the denser networks depreciate the model performance not only for the high degree vertices but also for the lower degree vertices. This is visible in Figure 2.6 for all denser degree distributions like the last two log-normal and linear functions.

On the other hand, the model excels in mimicking the desired degree distribution of the functions with a deep slope like the heavy-tailed functions and the steepest linear and log-normal functions. For such cases the correlation values where higher than 0.9 which denotes a near perfect match

This limitation of the BTER model can be easily been unnoticed since real-world networks tend to be sparse. Therefore, the model should be capable of generating networks similar to any degree distribution shown in the real-world scenarios.

I have also analyzed the ability of the model to match the desired clustering coefficients set in the parameters. I have found out that the final clustering coefficients are ultimately driven by the degree distributions and not by the parameter setup in the model. There are two important details to highlight:

(i) The quantity of degree one vertices;

(ii) And, the network connectivity.

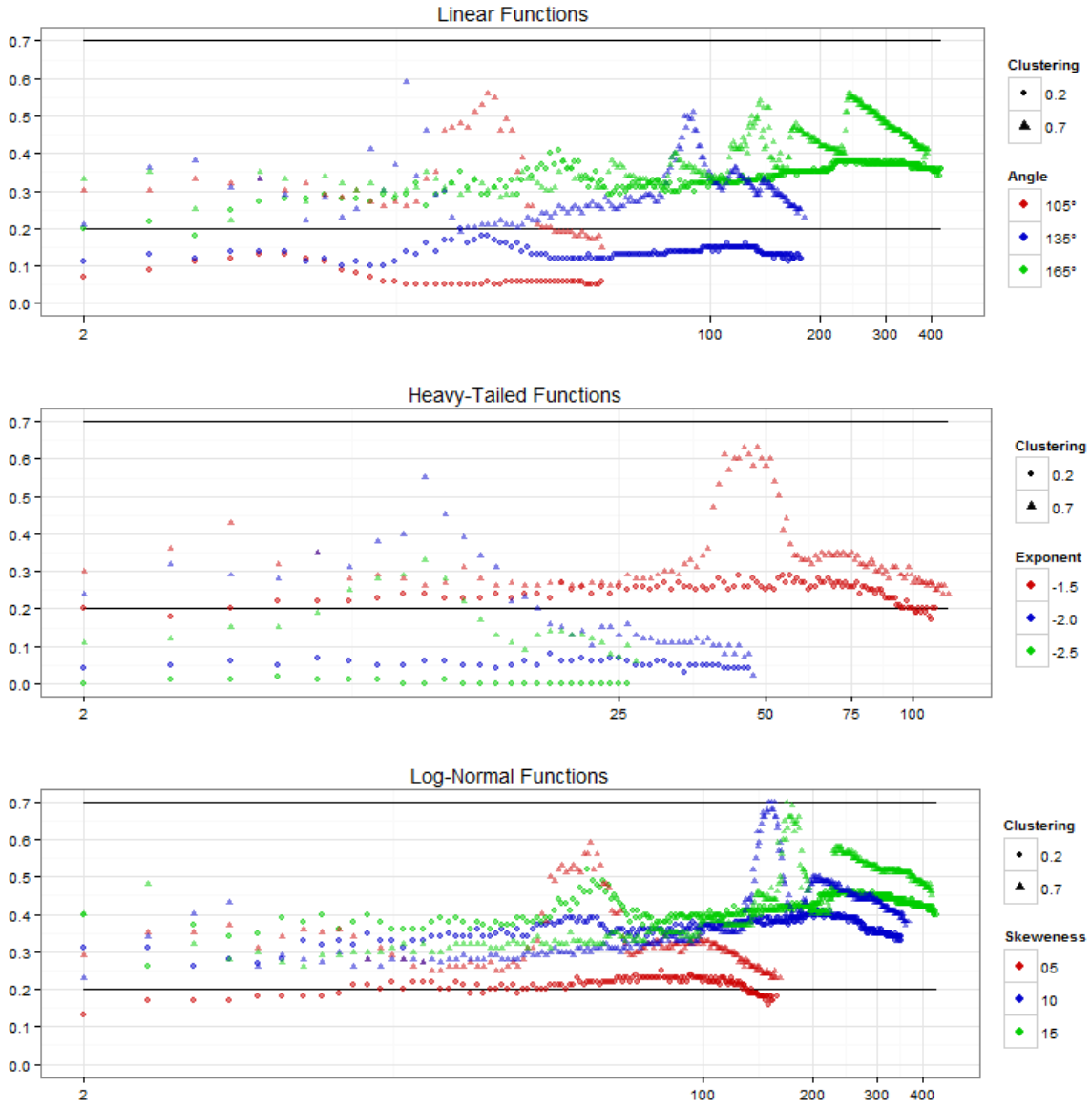
The large number of vertices with degree one (present in all degree distributions tested) and the restriction that grants connectivity to the generated networks forced that these degree-one vertices degenerate the clustering coefficients of many other vertices. They affect, in particular, the lower degree ones which are mainly responsible for the value of the global clustering coefficient of the network. This way, the degree distributions that present the larger number of degree-one vertices, and also generate the sparser networks, prevented high global clustering coefficients to happen.

The easiest solution for that problem is to delete the small components generated by the model (most of them formed by a solo vertex). Thereby, it was hard to control the final size of the network being generated. Another solution is to configure the clustering coefficient for each group of vertices with similar degree. This is an option of the model that enables more control over the connections and more stability than the global clustering. However, this option is costlier than the basic use of a global clustering because it requires a more complex tuning of the parameters.

To visualize the effect of the degree distribution over the clustering coefficients parameters I designed the graphs showed in Figure 2.7. Figure 2.7 presents the mean local clustering coefficients by degree of networks with 1,000 vertices. The triangles represent the networks where the desired clustering coefficient was set as 0.7 and for the points it was set as 0.2.

We can see clearly that the model has difficulties to fit the desired clustering value and that the main difference in the achieved clustering coefficients is caused by the degree distribution. This result suggests that one should configure the clustering coefficients by communities (vertices with same degree) instead of just inform the global clustering of the network. In that way, configuring the lower degree vertices (usually with smaller clustering) and the high degree vertices (higher clustering) separately will help the model to achieve the desired global clustering of the network.

Figure 2.7 – Local Clustering Coefficients by Vertex Degree



This fact mainly happens as a consequence of the way that the model generates the communities in the first step of the algorithm. Whenever it is unable to group vertices with equal degree, the model picks vertices with close degree to fill the group, but still considers the group to be formed exclusively with equal degree vertices for the second step of the algorithm. Thus, the lowest degree vertex of the community will achieve many of its connections inside the group and so a high clustering coefficient, while the other vertices inside the community will not.

Considering that most of the degree distributions present low number of vertices with high degree, it is predictable that the model will create only blended communities near the end;

in that way, the model generates peaks in the clustering coefficients towards the end of the degree distribution

In resume, I examined several different parameter setups and then show that the BTER is capable of matching various degree distributions. However, BTER is not capable to correspond to the desired clustering coefficients when restrictions such as connectivity are added to a given network. Further, I also show that the degree distribution plays an important role in the clustering coefficients, independently of how they are parameterized in the model.

The study of complex network models has played an important role in understanding real networks and the development of better algorithms. I have analyzed the generative characteristics of the BTER complex network model and, ultimately, I have shown that the BTER model is a good choice to generate synthetic networks similar to real-world environments.

In addition, I provided a way to set the parameters of the model to generate generic networks with custom-made degree distributions. Moreover, I demonstrated that the model has the ability to create a large variety of networks with distinct properties, which makes it a truly flexible model for analysis and simulations of large-scale networks.

In Table 2.5 I present a summary of the characteristics of the complex network models discussed in section 2. It shows the expected degree distribution and the assortativity coefficient[1] of vertices degree in the networks generated with each model. I also present the number of configurable parameters of each model and the overall complexity to configure such parameters to generate networks with desired characteristics (considering the restrictions and capabilities of each model).

[1] The assortativity coefficient or assortative mixing is the Pearson correlation coefficient of the degree between pairs of connected vertices. A positive assortativity means that vertices tend to be connected with vertices with similar degree while a negative coefficient means that vertices are connected with higher or lower degree vertices.

TABLE 2.5 – Complex Network Models Summary

Network Model	Degree Distribution	Assortativity	Number of Parameters	Configuration Complexity
Erdős and Rényi	Poisson	Positive	2	Low
Small-World Model of Watts and Strogatz	Dirac Delta	Positive	3	Low
Scale-free Networks	Power Law	Positive	3	Low
Networks with Community Structure	Multinomial	Negative	At least 3	Medium
Geographical Models	Poisson	Negative	At least 2	Low
Kronecker Graphs	Log-normal	Any	At least 5	High
Block Two-Level Erdős and Rényi	Multinomial	Any	At least 3	Medium

3 CENTRALITY MEASURES

Both the description and categorization of natural and human-made structures using complex networks lead to the important question of how to choose the most appropriate metrics and evaluations of structural properties. While such a choice should reflect specific interests and applications, unfortunately there is no general model, formal procedure or methodology for identifying the best measurements for a given network (COSTA et al., 2008).

In addition, the large number of metrics and their respective variations are often related despite the fact that each one of them consider distinct ideas for measuring graph properties. Ultimately, one has to rely on unwarranted intuition or limited knowledge about the problem to decide which metric is the most suitable for an application and to interpret it properly (COSTA et al., 2008).

We recall that networks are usually modeled by graphs where each vertex represents a node of the network and each edge may represent any kind of relationship between such nodes.

The overall structure of a network has consequences not only over individual members, but also over the entire group. Furthermore, structural properties of a network may extend well beyond individual behaviors and social roles. Assessing the quality of relations between entities and understanding connection patterns has generated much interest and research in various disciplines (COSTA et al., 2008).

Centrality measures can be characterized by deterministic algorithms (and associated formulations) that describe the computation of a "centrality" value for each vertex of a graph or network, usually creating a rank of vertices. However, the concept of what centrality is and what exactly it measures depends on its mathematical formulation and the application domain under analysis.

Nonetheless, there is no widely accepted formal definition of centrality in the context of social complex networks. As a result, there are several centrality measures capturing distinct ideas of centrality. In this context it may stand for importance, authority, power, influence, control, visibility, independency and/or contribution. (DAS et al., 2018; FREEMAN, 1978/79; LANDHERR et al., 2010).

Informally, the main question that the centrality measures try and answer is how central a vertex is, i.e. how much it is at the core or in the periphery of a network. Depending on the definition of centrality, an individual with high centrality is the one that has more control, visibility, or independence inside a network. Therefore, these measures are important to identify elements behavior and roles within a network (YU et al., 2017).

Vertex centrality measures make use only of the structural pattern of the network (i.e., vertices connections, edges weights, and temporal timeline) for their evaluation. There are metrics suitable for directed and weighted graphs, while others are used in strong connected simple graphs, or in temporal graphs. Consequently, all formulas and algorithms receive as input an adjacency matrix or adjacency list, with weights and temporal sequence when appropriate, representing the network under analysis (DAS et al., 2018; FREEMAN, 1978/79; LANDHERR et al., 2010).

For certain centrality measures, meta-parameters can be tuned to adapt the metric to a given kind of network structure or to control the weights given to different aspects/parts of the centrality. The metrics adapted to directed graphs are examples of the former case and the ones composed of more than one centrality measure are examples of the latter case.

Notice that in most applications where centrality is important, the vertices rank is used instead of their absolute centrality value. This aims at better comprehension, interpretation, and comparison of values. Rank centrality is a post-processing of the centrality measures where, for each vertex, a rank is given based on their centrality value decreasing order. If vertices are tied, their resulting rank is averaged.

Some works have evaluated important characteristics with respect to the use of existing metrics, see e.g. (BORGATTI et al., 2006; BUTTS, 2006; GOH et al., 2003; LI et la., 2015; VALENTE et al., 2008). Unfortunately, the relevance of their work in real-world applications is restricted to specific domains as they use a small number of experimental samples and centrality measures or restrict their analysis to specific kinds of applications.

It is important to highlight that there is no formal procedure to guide the choice of centrality measures for a given application.

In this section, I detail four of the main vertex centrality measures: betweenness, closeness, degree, and eigenvector. They are the most widely applied centrality measures in several applications and are the basis for several other metrics. I also briefly explain some of the most relevant metrics in the literature. Many of these centralities are very similar to each other in their conception and computed results and/or are adaptations of previous metrics, while others are specifically designed to networks in a determined field.

I focus on unweighted symmetric networks, so I present the metrics' version adapted for such kind of graphs but most metrics have simple modifications, at the expense of a higher computational cost (usually an order higher in the degree of the polynomial complexity), which turn their use suitable for most kinds of graphs. Table 3.5 at the end of this Section present more information about the complexity of such measures.

Centrality measures can be classified into four groups with respect to their purpose and conceptualization.

3.1 Degree Centralities

These are the simplest and most straightforward centrality measures. They were introduced by Shaw (1954), formally defined by Nieminen (1974) and popularized by Freeman (1978/79). These centralities are related to the idea of visibility and communication activity that a vertex has among its neighbors.

The degree centrality (CD) of a vertex w is the number of edges connected to it, i.e., the number of its adjacencies (1) (GRANDO and LAMB, 2016).

$$C_D\ (w) = \sum_{i=1}^{n} a\ (i, w) \qquad (1)$$

Considering a graph with n vertices and m edges, the corresponding algorithm has time complexity $\Theta(m+n)$ with an adjacency list (especially beneficial for sparse graphs) or $\Theta(n^2)$ with an adjacency matrix (usually preferable for denser graphs).

The metric is subdivided into indegree and outdegree for directed graphs. There are also variations in which the weights of the edges are accounted for.

The degree centrality is the least complex and has the smallest computational cost among centrality measures. Consequently, the overall class of metrics based on the degree are simple and easy to compute to any kind of graphs because it is the only class that considers only local information (i.e., it did not require information from the entire network structure only from its immediate neighborhood) to evaluate each vertex centrality.

However, these metrics has low capability at distinguishing vertices or ranking them, i.e., it considers many vertices as equally central. Notwithstanding, it is usually highly correlated to all other centrality measures despite its simpler formulation.

3.2 Path Centralities

This group of centralities evaluates the vertices as being central if they are in between (or at the "crossroads") of many "paths". This fact allows the vertices to control the communication through such paths and measures the frequency of arrival that an information piece diffused in the network trespass the vertex. Each centrality of this class considers different kinds of paths or consists of a distinct evaluation of these paths.

Most of these metrics require the graph to be strongly connected or evaluate each connected component of the graph individually and independently. However, there are more tolerant variations or adaptations that relax these restrictions to any kind of graph structure.

The most widespread and used metric is the betweenness centrality (CB), which considers only the smallest paths, called geodesics. The betweenness of a vertex *w* is the number of geodesics between vertices *i* and *j* that passes through vertex w divided by the total number of geodesics between the vertices *i* and *j* (2). Considering *i* and *j* all vertices of the graph, and *j* larger than *i* (a geodesic from *i* to *j* and from *j* to *i* is considered only once). This concept was introduced by Shaw (1954) and formally defined by Freeman (1977).

$$C_B(w) = \sum_{i=1}^{n} \sum_{j=i+1}^{n} \frac{g_{ij}(w)}{g_{ij}} \qquad (2)$$

The metric can be calculated using Brandes' algorithm (BRANDES, 2001), keeping its time complexity at O(*mn*) and does not need any specific representation for the input graph. The betweenness centrality of a vertex is the least complex metric in this class of centrality measures.

Other examples of centrality measures that belong this class of metrics are:

- Flow betweenness (FREEMAN et al., 1991): based on the network's maximum flow paths;
- Random walk betweenness (NEWMAN, 2005): it considers all possible distinct paths, but weights them according to the probability in which a random walk would pass through it;
- Communicability betweenness (ESTRADA, 2009): it considers all possible distinct paths, but rates the longest paths as less important;

- Range limited betweenness (ERCSEY-RAVASZ et al., 2012): considers only the shortest paths inside a range (length) limit.

3.3 Proximity Centralities

The basic idea of proximity centralities is that the lower the distance between a vertex to the others, the higher its centrality value and its independence from the network and its efficiency in propagating and spreading information. It also measures the time-until-arrival to the vertex of information pieces broadcasted through the network.

The main difference among these centralities is that each metric computes the "distance" between vertices in a distinct way. Since these centralities are based on distance metrics, there is an inherent problem with disconnected graphs: depending on the centrality measure, the distance between two disconnected vertices are considered infinite or the largest possible distance for the given network size.

The most prevalent and used centrality measure of this group is the closeness centrality (CC), first presented by Bavelas (1948) and then formally defined by Sabidussi (1966). Closeness centrality is the sum of the geodesics inverse distances from the vertex analyzed to all other vertices (3).

$$C_C\,(w) = \frac{1}{\sum_{i=1}^{n} d\,(i, w)} \qquad (3)$$

This metric can be calculated using a small variation of Brandes' algorithm (2008), keeping the same time complexity of betweenness centrality – and it does not require any specific representation for a graph. Closeness and betweenness can be computed simultaneously by merging both algorithms and by maintaining the same asymptotic time complexity, sparing computation resources in practice.

Other examples of centrality measures that belong to this class are:

- Informational centrality (STEPHENSON and ZELEN, 1989): it is based on the computation of the probability that a random walk starting from the start point ends in the target point;

- Eccentricity (HAGE and HARARY, 1995): very similar to the closeness centrality but considers only the distance to the farther vertex from the starting point instead of the distance to all vertices;
- Random walk closeness (NOH and RIEGER, 2004): the distance is measured by the average random walk time it takes to arrive to a target, so it leverages all lengths and possible distinct paths;
- Hierarchical closeness (TRAN and KWON, 2014): it is computed by a composition of closeness centrality and degree centrality (out degree in directed graphs) building a layer of a hierarchy at each iteration.

3.4 Spectral Centralities

All metrics that belong to this group evaluate the vertices centrality by their participation in substructures of the network. These measures capture the idea of involvement and contribution capabilities of a vertex to the network because it leverages its influence over the whole network structure. They are called spectral measures because of their relation with the set of eigenvalues of the adjacency or Laplacian matrix of the graph representing the network.

Perhaps the mostly widely used among these measures is the eigenvector centrality (CE). Bonacich (1972) suggested the centrality based on the eigenvector of the largest eigenvalue of a network's adjacency matrix. Eigenvectors can be seen as a weighted sum of not only immediate contacts but, as well as, indirect connections with every vertex of the network of every length. Moreover, it weighs contacts of a vertex according to their own centrality, i.e., links with central vertices contribute towards their own centrality ("the more powerful your friends are the more powerful you become").

The eigenvector respective to the largest eigenvalue can be computed via an iterative procedure known as "power method" using the adjacency matrix and an auxiliary vector (RICHARDS and SEARY, 2000), which reduces its computational cost considerably and avoids numeric precision issues. The power method requires an infinite number of steps (worst case) but as the number of steps increases, the precision of the measure also increases. Therefore, the number of decimal places can be used to turn this measure feasible even for massive networks where a hundred steps usually grants enough precision to differentiate the vertices centrality values (RICHARDS and SEARY, 2000).

The eigenvector centrality value of a vertex *w* at an iteration it is the *w* index of a vector E multiplied by the adjacency matrix *A* divided by the sum of the elements of *E* at a previous iteration (4). The vector *E* can be initialized with any real positive number (RICHARDS and SEARY, 2000).

$$C_E\ (w) = \sum_{it=1}^{+\infty} \frac{(E^{it}A)_w}{\sum_{i=1}^{n} E_i^{it-1}} \tag{4}$$

The following centrality measures also belong to this class of metrics:

- Katz centrality (KATZ, 1953): similar to the eigenvector centrality, but uses an attenuation factor to reduce the influence of distant vertices;
- PageRank centrality (PAGE et al., 1999): similar to the Katz centrality, but adds a dumping factor which enables its proper use in directed graphs (correcting the rank sink and drain problems caused, respectively, by cycles and vertices with zero outdegree);
- Subgraph centrality (ESTRADA and RODRÍGUEZ-VELÁZQUEZ, 2005): evaluates the vertices by their participation in subgraph structures of the network, starting with subgraphs composed by only two vertices and increasing in size to any possible formation with unlimited bounds, but gives less weight as the subgraph becomes larger;
- Functional centrality (RODRÍGUEZ-VELÁZQUEZ et al., 2007): similar to the subgraph centrality, but with limited range on subgraph structures.

3.5 Centrality Measures Comparison

I carried out a set of experiments with eight of the most used centralities to compare its characteristics in several networks. The selected metrics were closeness (C_c), betweenness (C_b), degree (C_d), eigenvector (C_e), information (C_i), subgraph (C_s) and eccentricity (C_x).

I have chosen six complex network models with a hundred samples for each possible combination of the selected parameters (presented in Table 3.1). The parameters were chosen following a preliminary experimental setup that verified the main properties of the networks generated by each model. The network models were used to provide several samples to achieve a good generalization of my hypotheses and results so that they were statistically relevant. For

more information about these complex network models, comprising their main characteristics and explanations about their parameters, please see section 2.

TABLE 3.1 – Complex Network Models Parameters Summary

Network Model	Parameters in Experiments
Networks with Community Structure (M_{cs})	p_c = probability of each vertex to belongs to each community {0.1}. p = edge probability between two vertices belonging to a common community {0.5, 0.7}. c = number of communities {n/10, n/20, n/50}. n = number of vertices {100, 500}.
Simple Random Graphs (M_{er})	p = probability of connecting each pair of vertices {0.1, 0.3, 0.5}. n = number of vertices {100, 500}.
Geographical Models (M_{gr})	p_{ij} = probability that vertices i and j are connected $\{k^{-s_{ij}}\}$. s_{ij} = distance between vertices i and j $\{\left\|\left\lfloor\frac{i}{\sqrt{n}}\right\rfloor - \left\lfloor\frac{j}{\sqrt{n}}\right\rfloor\right\| + \left\|(i \bmod \sqrt{n}) - (j \bmod \sqrt{n})\right\|\}$. k = variable used in the p_{ij} equation {1.2, 1.5, 2}. n = number of vertices organized in a determined space {100, 500}.
Scale-Free Networks (M_{sf})	k = initial number of fully connected vertices and the number of edges added with each new vertex {2, 3, 5}. n = final number of vertices {100, 500}.
Small-World Model (M_{sw})	p = probability of changing each connection (relinking) {0.1, 0.3, 0.5}. k = initial number of nearest neighbors which a vertex is connected {4, 8, 16}. n = number of vertices connected in a ring structure {100, 500}.
Kronecker Graphs (M_{kg})	P = square matrix of order two with parameters estimated for different kinds of networks: social {Email-Inside, Epinions}, information/citation {Blog-Nat06All}, Web {Web-Notredame}, Internet {As-Newman, As-RouteViews} and biological {Bio-Proteins}. n = number of vertices {2^7=128, 2^9=512}.

I have also used non-isomorphic[2] connected graphs (Nni). I selected all Nni of six (112 graphs) and seven vertices (853 graphs) retrieved from a public dataset available at http://cs.anu.edu.au/~bdm/~data/graphs.html. These graphs are useful to illustrate possible extreme configurations for centrality values. Some of these are very unlikely in random generative models. The sizes (six and seven vertices) were chosen because they allow variability and they can be subject to experimental analyses (e.g., there is a total of 11,716,571 non-isomorphic connected graphs of ten vertices).

[2] Two graphs are non-isomorphic if there is no possible edge permutation that transform a given graph into the other.

The final experimental setup contained a total of 7,165 synthetic networks (accounting the networks generated with the six complex network models and also the non-isomorphic graphs).

The analysis of my experiments starts with the centrality measures' correlation to check their similarity or distinctiveness. The comparison between the measured centrality values via different centrality measures can unveil relevant information about a network structure. It can assess a distinct role for a given vertex in a network considering that each metric gives a different meaning of vertex importance. Therefore, the correlation among metrics gives an idea of the relation between such distinct roles. I used the Kendall Tau-b rank correlation coefficient between every combination of metrics to estimate the metrics' relationship. This coefficient evaluates the degree of similarity between two sets of ranks given to the same set of objects. Kendall's correlation is especially useful for centrality measures because varying normalization and distribution do not affect it. In addition, centrality measure values are used frequently as ranking factors where the absolute value itself is irrelevant. The same analytical approach was used by other works in similar areas (BAIG and AKOGLU, 2015; SOUNDARAJAN et al., 2014).

The mean Kendall rank correlation coefficients between each pair of metrics are summarized in Table 3.2. All values presented in the table match the real expected value with 0.01 confidence interval (above and below) with 99% chance. The correlation values above or equal 0.8 are highlighted in bold.

TABLE 3.2 – Mean Correlation Values

C_c							
0.75	C_b						
0.79	0.78	C_d					
0.80	0.63	0.79	C_e				
0.83	0.77	**0.91**	0.79	C_i			
0.76	0.61	**0.80**	**0.94**	0.77	C_s		
0.73	**0.87**	**0.82**	0.63	0.79	0.62	C_w	
0.42	0.35	0.33	0.35	0.36	0.32	0.33	C_x

In summary, all metrics (except eccentricity) presented a high redundancy in their evaluation of vertex centrality by showing a high correlation with the other metrics. The pairs of metrics closer to each other in decreasing mean correlation value order (considering all networks) were: eigenvector and subgraph (0.94), degree and information (0.91), betweenness

and walk betweenness (0.87), closeness and information (0.83). It is noticeable that eccentricity lags behind, with correlation values below 0.45 in the great majority of networks.

We see that just a few pairs of centralities have low correlation (excluding those with eccentricity), which means that despite all centrality differences, they all have a high common agreement when ranking vertices of a network by their centrality in all kinds of networks tested. Eccentricity presented the lowest correlation values by far, mainly because it evaluates many vertices as being equally central. Notice that the Kendall tau-b rank correlation accounts positively for pairs of tied values only if both sequences present the same elements with equivalent ranks, penalizing them otherwise.

Eigenvector and subgraph centralities acquired a perfect correlation coefficient (1.00) in most networks but, at the same time, they presented low correlation in some rare cases (below 0.5). This emphasizes that both of them are so similar that choosing one metric over the other does not have any impact in the results. It also means that the idea of centrality behind each metric are equivalent in a practical sense.

We also notice that the correlation values among metrics vary according to the generative model. In the networks generated by M_{sw} and by M_{sf}, the overall correlation values for all metrics were much lower (23% and 20% below the mean respectively) while in the networks generated by the M_{kg} and by M_{er}, they were considerably higher (19% above the mean).

The networks generated by the other two models presented correlation values close to the mean. This fact is consistent with the degree variability of each model i.e., the larger is the range of the degree distribution generated by the model the lower is the correlation among centrality measures.

The only metric that behaved differently was eccentricity. This happened due to a large number of tied centrality values in the M_{er} networks. The lower the diameter of a network, the lower is the variety of distances among vertices and therefore the granularity of eccentricity. This fact penalizes its correlation values with all other centrality measures, granting it a 66% lower correlation than the average value over all networks.

The correlation among centrality measures varies, as expected, in different setups of parameters (including networks size) for a complex network model, but their variance was statically irrelevant. That is mainly because the chosen parameters preserved the fundamental characteristics of each complex model.

Furthermore, I showed that the structural properties of the networks do not significantly affect the centralities granularity while they do have a considerable impact on their correlation.

These results are a strong indicative that the simultaneous use of some pairs of metrics is quite redundant and will provide limited results. Therefore, the simpler metrics, such as betweenness, closeness, degree and eigenvector, are preferable and suitable for most applications, being easier to interpret and having lower computational complexity.

The second property of the measures that I analyzed was their granularity, i.e., their capacity to distinguish and evaluates differently, providing an exclusive centrality value for all the vertices in the network considering that they represent unique elements of a real-world environment. I calculated the percentage of distinct centrality values for each metric in each network. For example, if I say that there are 50% distinct values in a network of 500 vertices, it means that there are 250 distinct (unique) centrality values for a given metric in that network. I considered six decimal places (rounded) as the accuracy for the real value centralities as enough to distinguish the centrality values properly, considering the size of the networks used in my experiments. This was necessary to avoid numeric precision issues common during the computation of the metrics with the inherent imprecision of the operations and memory storage of floating-points format variables (Double for instance). They are used during the computation of the metrics as they are much less computational costly than the operations with precise decimal variables. The granularity property may be valuable in applications where centrality measures are used to differentiate/rank the vertices of the network (ŻAK and ZBIEG, 2014) or as a heuristic for vertex selection and placement (MARCHANT et al., 2015).

Tied values in these applications may suggest many unequal solutions for a given task in which only a best solution is required or desired. In addition, ties between centrality values of distinct vertices can be viewed as lack of information or incapability of the metric to differentiate the vertices properly, considering the fact that they are definitely unique in many domains. Moreover, correlation analysis between centrality measures and other domain-specific metrics are common. Such methods are impacted by equivalent values, reducing their accuracy and distorting the analysis.

Table 3.3 presents the mean percentage of distinct values with 99% confidence intervals for each metric, grouped by the networks generated by the complex network models and by the non-isomorphic networks. The results show that the granularity of all centrality measures presented nearly no difference in the expected values (considering the confidence intervals) in all networks generated by the complex network models. This is why I put them altogether in Table 3.3

TABLE 3.3 – Mean Percentage of Distinct Values

Metric	Complex Models (%)	Non-Isomorphic (%)
C_b	98.72 ± 0.11	59.13 ± 1.72
C_c	39.34 ± 0.60	55.07 ± 1.45
C_d	19.11 ± 0.51	48.93 ± 1.12
C_x	1.36 ± 0.04	26.86 ± 0.83
C_e	99.57 ± 0.03	70.54 ± 1.92
C_i	99.84 ± 0.02	69.36 ± 1.90
C_s	94.15 ± 0.26	69.19 ± 2.00
C_w	99.68 ± 0.04	69.56 ± 1.87

Eccentricity underperforms all other metrics with a high number of tied vertices due to its simple formulation. It is followed by degree and then by closeness centralities in all synthetic networks. These three measures are by far worse than the others are in distinguishing vertices by their structural properties.

Walk betweenness, information and eigenvector are the metrics with the best granularity. They are followed narrowly by betweenness and then by subgraph centralities, forming a group with five high granularity metrics. This fact that is supported by their more complex formulations compared to the other three metrics.

Our results provide additional evidence that the degree measure is less fine-grained than closeness and both are inferior to betweenness in this aspect, as Freeman (1978/79) originally thought but did not presented empirical results as I did. The intuition behind this fact is mainly because the simpler way that these three metrics are evaluated.

Other interesting aspect of centralities analyzed in my experiments was the number of times each metric achieved the best-known granularity solution among the metrics tested. This information shows the number of times that one metric is better than all others are in distinguishing the vertices of a given network.

The results are summarized in Table 3.4. The cells highlighted in bold present the highest values. Notice that the columns total is higher than 100% because in many networks more than one metric achieves the best/top granularity performance. Table 3.4 reinforces even more the disparity in metrics granularity. The top overall metric walk betweenness is better in this aspect than all others are in most networks, excluding geographic and Kronecker networks, where information centrality performs well, and in non-isomorphic networks, where eigenvector is the best-qualified one.

TABLE 3.4 – Percentage of Times with Best Granularity

Metric	N_{ni}	M_{cs}	M_{sf}	M_{sw}	M_{gr}	M_{er}	M_{kg}
C_b	38.8%	97.6%	62.8%	78.3%	70.2%	**100%**	10.9%
C_c	33.7%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
C_d	21.5%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
C_x	4.9%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
C_e	**98.4%**	87.7%	51.7%	47.9%	62.0%	55.0%	82.5%
C_i	90.6%	98.0%	92.8%	90.6%	**98.8%**	67.3%	**94.8%**
C_s	93.4%	32.4%	31.3%	34.2%	34.3%	32.0%	21.3%
C_w	92.0%	**99.8%**	**100%**	**99.9%**	76.5%	**100%**	38.6%

It also became evident the poor granularity of eccentricity, degree and closeness (in increasing order from the poorest one) centralities compared to the others. The only networks where their granularity is at least equivalent to the other metrics were special cases of non-isomorphic networks, such as, the complete graph, when all centrality measures evaluate all vertices as being equally important.

The higher granularity of the more complex measures seems to be the fundamental reason to use them in any application. Despite proving to be highly correlated to the simpler measures, they proved to be more able to differentiate the vertices of a network. The ability to distinguish and rank the vertices of a given network, preferably without tied elements, is very important for some applications.

In Table 3.5 I present a summary of the characteristics of the centrality measures discussed in this section. It summarizes relevant properties of each centrality measure and may help one to choose the most appropriate metric for a given application. The following characteristics are depicted in the table:

- If the metric's algorithm can be applied to digraphs and weighted graphs (a * sign in these columns' cells notifies that there is a more complex version of the metric's algorithm that supports such characteristic or type of graphs);
- The expected parallelism speedup due to algorithmic restrictions and dependencies considering that the centrality measure will be computed in parallel for each vertex of the analyzed network (a high parallelism speedup means that the centrality can be computed almost independently for each vertex with nearly no dependency from the other parallel threads, while in a medium speedup there are some dependencies and at low speedup the parallelism is close to ineffective due to the amount of dependencies);
- The granularity of each measure, i.e., its ability to distinctly evaluate the vertices;

- The existence of manual parameters that have to be tuned considering the purpose and size of each kind of application;
- The time complexity of the best-known centrality algorithm considering the asymptotic bounds and a graph with n vertices, m edges, diameter d and l is used to define the range parameter.

It is important to notice that although Eigenvector, Katz and PageRank exact algorithms have quadratic upper bounds considering their time complexity, they perform in practice near linear time due to the application of an iterative version of the algorithms ("*power method*") being the most efficient metrics in terms of computational costs after the class of degree centrality metrics. For instance, this fact enabled the use of such centralities in the core of the ranking system of Web search engines, such as Google (PAGE et al., 1999; RICHARDS and SEARY, 2000).

TABLE 3.5 – Centrality Measures Summary

Centrality Measure	Digraph	Weighted Graph	Parallelism Speedup	Granularity	Param.	Complexity
Degree	No	Yes	High	Low	No	$\Theta(m)$
Indegree	Yes	Yes	High	Low	No	$\Theta(m)$
Outdegree	Yes	Yes	High	Low	No	$\Theta(m)$
Betweenness	No*	No*	High	High	No	$O(mn)$
Flow Betweenness	No*	Yes	Medium	Low	No	$O(m^2n)$
Random Walk Betweenness	No	No	Low	High	No	$O(mn^2+n^3)$
Communicability	No	No	Low	High	No	$O(n^4)$
Range Limited Betweenness	Yes	Yes	Low	High	Yes	$O(mn^{l+l/d})$
Closeness	No*	No*	High	Medium	No	$O(mn)$
Information	No	Yes	Low	High	No	$O(n^3)$
Eccentricity	No*	No*	High	Low	No	$O(mn)$
Random Walk Closeness	No	No	Medium	High	Yes	$O(n^2)$
Hierarchical Closeness	Yes	No*	High	Medium	No	$O(mn)$
Eigenvector	No	No	Medium	High	Yes	$O(n^2)$
Katz	Yes	No	Medium	High	Yes	$O(n^2)$
PageRank	Yes	Yes	Medium	High	Yes	$O(n^2)$
Subgraph	No	No	Medium	High	No	$O(n^2)$
Functional	No	No	Medium	High	Yes	$O(n^2)$

3.6 Applications of Centrality Measures

Centrality measures are used as important analyses tools for graphs and are used in several applications in which networks are fundamental. They are usually employed as a mean to identify relevant and important elements by identifying their behavior and roles within given complex networks or to rank such elements by their contribution for such roles which is important to guide selection mechanisms in numerous situations. Centrality measures are also effective as a comparison factor with other domain-specific metrics and can be used as a bridge to other areas (BRANDES, 2001; FREEMAN, 1977; GRANDO et al., 2016).

In the coming subsections, I highlight promising areas in which recent studies have used centrality measures as a meaningful instrument. I do so by a simple taxonomy, classifying such examples considering their main application areas – of course, some of the researches are related to more than one application domain.

3.6.1 Computer Networks

Given the ubiquity of computer networks structures, the use of centrality measures is pertinent as heuristics to improve the solution of security problems, control issues, communication flow, and resources optimization. One of many examples is the study of Maccari and Cigno (2016), who applied the betweenness centrality in the optimization of online routing control protocols responsible to provide a fast and efficient recovery from a node failure with minimal disruption of routes. They consider that the failure of a node with high centrality (they have focused in the top 10 ranked vertices) generates higher loss compared to the failure of peripheral nodes and applied this fact in their proposed optimization problem formula to minimize routes' disruption with reduced control message overhead. They also pointed it out that, although the centrality needs to be computed online (it takes about seconds to compute) and this is feasible only in networks with hundreds of vertices, there are approximation techniques that may be used for networks with thousands or more vertices. Moreover, the same technique can be extended to distance-vector protocols that are largely used in wireless mesh networks (VÁZQUEZ-RODAS and LLOPIS, 2015).

Kas et al. (2012) studied social network analysis metrics and centrality measures as tools to improve the design of wireless mesh networks with better performance by efficiently allocating the resources available.

The allocation of network resources using the centrality measures as heuristics has been a topic of study in distributed placement of autonomic Internet services for the efficient support of users' demands (PANTAZOPOULOS et al., 2014).

Ding and Lu (2015) studied nodes' importance (given by centrality measures) to maintain the structural controllability of a network. They introduced the control backbone for quantifying a single vertex's contribution to the network controllability based on a random sampling algorithm that uses centrality measures as basis.

Other key application of centrality measures is in wireless community networks (WCNs). WCNs are bottom-up broadband networks empowering people with their on-line communication means (BALDESI et al., 2015). However, WCNs often lack in performance because services characteristics are not specifically tailored for such networks. Centrality measures are used as heuristics that improve the selection of peers and chunks within WCNs in the communication between users to reduce the impact of P2P streaming while maintaining applications performance.

There are also studies about mobile and 5G wireless networks that used centrality measures to reduce network traffic (BAŞTUĞ et al., 2014a; BAŞTUĞ et al., 2014b).

Likewise, centrality measures are studied in the context of network security. For instance, they have been used to configure distributed firewalls (MACCARI and CIGNO, 2013) and to identify critical nodes in which intrusion detection and firewalling is most suitable (MACCARI et al., 2016).

In addition, they were applied to build cloud-based filters and footprints of traffic inspection algorithms at global scrubbing centers (ZILBERMAN et al., 2015).

3.6.2 Complex Networks Construction

The analysis and understanding of the structure of complex networks are fundamental to understand variables and properties that contribute to the network formation and to identify which premises affect network development and evolution (EASLEY and KLEINBERG, 2010; CHEN et al., 2016). Centrality measures help these studies by identifying nuances and characteristics of components of the networks. In such context, centralities are used as means of understanding the role or contribution of a given element and its effect on the other elements of the entire network.

König et al. (2010) analyzed the underlying differences between technological and social networks and the assortativity behavior of network elements. They considered that the link formation is mainly based on the vertex centrality.

Centrality measures were also used as part of a dynamic network formation model to explain the observed nestedness in real-world networks (KÖNIG et al., 2014) and social networks community structures (NEWMAN and PARK, 2003).

They are also fundamental to several community detection and formation algorithms. The betweenness centrality measure was applied successfully by Newman and Girvan (2004) as a heuristic metric in order to define proper cuts/divisions in social networks as a way to separate the elements and iteratively search for communities.

The idea behind the algorithm proposed by the authors is that the more central elements are bridges between different communities. As an iterative algorithm, it needs to compute the centrality measure again at each step/cute/division made, and because of its frequent use in huge social networks an efficient implementation is required. Therefore, approximation methodologies are preferred most of the time.

3.6.3 Artificial Intelligence Applications

The use of network models in communication, cooperation, or learning structures is key in artificial intelligence (AI). Several AI domains offer themselves to the use of centrality measures as means of strategy, guidance, or simply as domain information. Machine learning algorithms and methods can use centrality measures as heuristics to learn faster, to synthesize input data, or as additional information about an AI application, which in turn helps in the generalization for a given domain.

Pedestrian detection is one of the example problems that used centrality measures associated with machine learning techniques. Detecting pedestrians in computer vision is a critical problem when one considers that the performance of trained detectors may drop very quickly when scenes vary significantly, which is frequently the case. Cao et al. (2013) proposed a novel methodology for such a problem based on a bag of visual words to detect pedestrians in unseen scenes by dynamically updating the keywords using centrality measures to select new keywords on the manifold model.

Other AI application used the metrics to classify label-dependent nodes in a network (e.g. hyperlinks connecting web pages), citations of scientific papers, e-mail conversations, and social interactions in the Web (KAZIEENKO and KAJDANOWICZ, 2012).

Centrality measures can be applied in several ways as an optimization heuristic in multi-agent systems and multi-robot teams, since they are essentially driven by their communication organization and strategies. Xu et al. (2014) noticed through simulations that the more central robots (considering its betweenness centrality) in a large robot team (organized frequently as clusters/networks) are the ones responsible to broadcast the information amongst clusters. Consequently, these central elements play an important role in group efficiency and are helpful to speed up the information diffusion.

In addition, reinforcement learning applied to agents' automatic skill acquisition and selection has also been studied using centrality measures (MORADI et al., 2010; RAD et al., 2010).

3.6.4 Social Network Analysis

There is a growing amount social networks data, from different sources. Therefore, it is crucial to study the available information, as many networks comprise hundreds of millions of vertices. Centrality measures are one of the most important analyses tools for these kinds of networks and can be used for many purposes and fundamental comparisons, which offer insights on their social impact and behavior.

Danowski and Cepela (2010) used centrality metrics to analyze the impact that presidential centrality role has on presidential job approval ratings. They hypothesized that when the centrality of the president is lower than of the other cabinet members, job approval ratings is higher.

Jiang et al. (2015) introduced a model to find influential agent groups and their impact in multiagent software systems by using centrality measures, while Mcauley and Leskovec (2012) used the metrics to search and identify social circles in ego networks.

The identification of important elements and an impact analysis of coautorship networks (YAN and DING, 2009) is also a research topic where centrality measures proved to be fundamental. They have used data from 16 journals in the field of library and information science with a time span of twenty years to construct a coautorship network and, they tested the use of four centrality measures (closeness, betweenness, degree and PageRank) as predictors of articles' impact. To validate such an analysis, they compared the results obtained with the centralities with the citation counts. They unveiled a high positive correlation amongst it and the metrics, which strongly suggests that central authors are mostly likely to have a better reputation and produce articles with increased citations.

Louati et al. (2014) use software agents to formally express and interpret semantic information useful to evaluate trust:

(i) Judging if the provider is worthwhile pursuing before using his services (trust in sociability);

(ii) Expert-base aspect such as estimating whether the service behaves well and as expected (trust in expertise);

(iii) And, recommender-based aspect such as assessing whether an agent is reliable and we can rely on its recommendations (trust in recommendations).

They applied the centrality measures in several parts of their proposed algorithm as selection heuristics to desired trustworthy services for social networks.

Likewise, there are studies about the setup of marketing strategies involving the selection of influential agents that increase the impact factor and improve the match between marketing content and users' expectation and interests have also proved the need for centrality measures (LI et al., 2016).

3.6.5 Traffic and Transport Flow

Physical transport of goods and people are a huge strategic logistic problem for cities, states, and countries. For instance, the analysis of terrestrial, air, and water traffic networks used centrality measures as heuristics to solve flow and routing problems. Centrality measures are also applied in several applications that considers social behavior and use GPS to record and analyze the traffic in real-time. These algorithms are fundamental to find better solutions for routing and car-sharing and take in consideration several factors to be efficient.

Gao et al. (2013) analyzed the potential of centrality to predict road traffic and flow patterns. They examined urban street flow using the taxis trajectory (recorded via GPS) and compared to the result predicted using betweenness centrality and the consideration of spatial heterogeneity of human activities, which was estimated using mobile phone Erlang values. The combination of centrality with other techniques showed to be extremely effective.

Hua et al. (2010) analyzed the United States air transportation network to unveil the importance and impact on the economy that each airport has over the air traffic network in each region and as a whole. They also used temporal analysis to compare with previous studies about such networks.

Centrality measures and their models are also used in global and local scenarios and on the analysis of different points of view (intersection, road, and community views) in such scenarios (JAYASINGHE et al., 2016; ZHAO and ZHAO, 2016).

3.6.6 Centrality in Game Theory

In game theory, centrality measures are employed in studies about coalitional games. Coalitional or cooperative games are usually formed by groups of individuals that are connected via a network and the efficiency of this network is often related to the groups' performance.

Noble et al. (2015) studied the impact and relationship between the centrality of an agent with their collective performance in the resolution of cooperative problems. They tested several centrality measures (betweenness, closeness, degree, and eigenvector) combined with different network structures (defining communication channels among agents) and distinct solving strategies (evolutionary algorithms) in a set of minimization problems (real-valued functions). They showed that the centrality of an agent severely affects its contribution for the overall collective performance of the network.

Centrality measures also have shown significance to define strategies for the dynamic formation and evolution of networks connecting individuals in social and economic game-based networks. Further, the measures have also been applied to risk assessment, considering fairness and efficiency, and to identify a bank's systemic impact and relevance (JACKSON and WATTS, 2014).

There are games that also make use of complex centrality measures, and as they require efficient algorithms, they sometimes apply approximation centrality algorithms to be viable (MICHALAK et al., 2013).

3.6.7 Biological Networks

Biological networks are examples of network structures not built by humans composed by biological structures. Protein-protein interaction (PPI) is one particular type of biological network where centrality measures have been applied.

XIONG et al. (2014) have applied centrality measures like betweenness, degree and closeness to select proper/informative candidates (proteins) for labelling in PPI networks clusters and then classify/predict their purpose or function inside a biological cell. They reason that the more central proteins in a PPI network cluster are probably responsible for a given cell

function. The identification and the behavior of important proteins related to specific reactions in a cell constitute a huge step towards the development of drugs with minimal collateral effects and also to better understand diseases and hormonal interactions and their effects. Note that PPI networks are usually huge, sparse, and numerous. So, an efficient analysis method is fundamental in these applications and frequently approximation algorithms for the centrality measures are preferable in such environments.

The metrics were also important in several studies about human brain connectivity and its behavioral interpretations. Neural connections and interactions (synapses) form huge complex networks where actions of a living being is controlled and determined. Therefore, the identification, classification, clustering, behavior, and inner-relationship are topics of high research interest. There is a growing demand for the applications of centrality measures in the biomedical sciences due to their successful use in many network related topics and the insights obtained from several application domains (RUBINOV and SPORNS, 2010; HEUVEL and SPORNS, 2013).

The next section will give a brief introduction on machine learning and associated algorithms which are used in this work’s experiments.

4 MACHINE LEARNING

Machine learning is a subarea of artificial intelligence (AI) that uses statistical, linear algebra and calculus techniques to give computers the ability to "*learn*" by progressively improving the performance of a model (output generated by the learning algorithm) on a specific task without being explicitly programmed for such task (MITCHELL, 1997).

Machine learning uses a variety of algorithms that iteratively learn from data to improve, describe data, and predict outcomes. As the algorithms consume training data, they produce more precise models based on that data (SEGARAN, 2010).

Some of the machine learning models are online and continuously adapt as new data is presented, while others are offline, i.e. once the model is deployed (after the training) it does not change (CONWAY and MYLES, 2012; HEATON, 2013).

There are many machine learning algorithms, techniques and methodologies but they all need some kind of training data/information about the environment or task/problem and, all share the same goal of creating an ideal model capable of simulating, resuming and simplifying, the environment. They are employed in a great range of tasks, including but not restricted to:

- Classification – the machine learning algorithm needs to produce a model capable of assigning the input data into predefined sets or classes. In most of these problems the model is trained with labeled data (supervised learning);
- Regression – different from the classification problems the model outputs/targets are continuous rather than discrete. The supervised learning is also the most common approach in this kind of tasks;
- Clustering – in these tasks a set of data is to be divided into groups that are not known beforehand as they are in the classification problems. Because the groups are not known, the training data is unlabeled (unsupervised learning), consequently, the most common approach is to group the data by a similarity criterion and by using statistical approaches;
- Reinforcement Learning – these tasks deal with dynamic settings where the model needs to learn by interacting with the environment (trial and error). The only information provided for the model in these tasks are usually sensors outputs by giving rewards/penalties depending on the outcomes for a given interaction with the environment;

- Temporal Learning – the input data in these problems, distinctly from the others, have time correlation, consequently, the order of the data is important. The main objective in such tasks are to create a model capable of predicting the future (expanding the timeline of the previous input data) or a model that explains and also recreates temporal patterns.

This set of algorithms and models are being used across industries to improve processes and gain insights into patterns and anomalies within data (HASTIE et al., 2017; BISHOP, 2006).

AI and machine learning algorithms are not new in the literature. This field dates back to the 1950s when Arthur Lee Samuels, an IBM researcher, developed one of the earliest machine learning programs for playing checkers and coined the term "*machine learning*" on a paper published in the IBM Journal of Research and Development in 1959 (RUSSELL and NORVIG, 2015).

Over the years there was a growing interest in machine learning techniques by researchers and industry and a huge amount of money are investments today in the area. The success of the area is due to several factors (BIRD et al., 2009; RUSSELL and NORVIG, 2015):

- Modern processors have become denser, cheaper and increasingly powerful;
- The cost of storing, managing and analyzing large amounts of data has been dramatically reduced;
- The ability to distribute compute processing across clusters has been improved;
- The availability, volume and variety of commercial data sets for studies increased;
- The machine learning algorithms have been implemented by open-source communities with large user bases which induced the creation availability of more resources, frameworks and libraries;
- The analysis, consume and visualization of data became more popular and accessible for industries and researchers.

All of this means that it is possible to produce, quickly and automatically, models capable of analyzing bigger and more complex data and, at the same time, deliver rapid and precise results even at grand scale.

Machine learning can be divided in two main classes: symbol-based and connectionist learning (LUGER, 2002). Symbol-based learning methods use a set of symbols that represent the entities and relationships of a problem domain and attempt to infer novel, valid, and useful generalization that can be expressed using these symbols (LUGER, 2002).

The connectionist approaches represent knowledge as patterns of activity in networks of small, individual processing units and learn by modifying their structure and weights in response to training data. Rather than searching through the possible generalization afforded by a symbolic representation, connectionist models recognize invariant patterns in data and represent these patterns within their own structure. Moreover, all patterns, symbols and results are encoded as numerical vectors (LUGER, 2002).

The next subsections present a summary about three classes of machine learning algorithms: decision trees (symbolic learning), support vector machines and artificial neural networks (connectionist learning). These algorithms are commonly applied to regression tasks and were tested in my experimentations to approximate the centrality measures (section 5.3).

4.1 Decision Trees

Decision trees are one of the most widely, practical and readable methods for inductive inference. It is able to deal with continuous and discrete data and it is also robust to noisy and inconsistent data (MITCHELL, 1997). Decision trees are the general name for this class of algorithms but they also can be specifically applied to classification problems and term regression trees when these algorithms, or their variations, are used in regression problems.

Decision tree algorithms creates a model in a tree structure to illustrate and map the possible results of a decision. They use the divide and conquer strategy to solve a complex decision problem that is subdivided in smaller problems recursively where each subspace can be adjusted by using different models and later combined to solve the full and more complex problem (FACELI et al., 2011).

It also can be represented as sets of if-then rules to improve readability. In general, decision trees represent a disjunction of conjunctions of constraints on the attribute values of instances. (MITCHELL, 1997).

Each node of a decision tree model specifies a test for some of the attributes of the instance and each descending branch from that node corresponds to one of the possible values for that attribute. An instance is classified by starting at the root node of the tree, testing the attribute specified by this node, then moving down the corresponding tree branch. The process is repeated in the subtree structure until you find a class label or regression value at a leaf node. It is important to highlight that in regression tasks, the output of the instances must be discretized first in a finite number of classes, ranges or categories.

There are plenty algorithms that implement a decision tree learning model. These algorithms are responsible to generate the tree structure by specifying the attributes testing order (starting with the best classifier at the root, i.e. the one that separates better the training samples according to the target values). For such task, the algorithms quantify the information gain that a determined division provides by measuring the expected reduction in entropy (measure of homogeneity of the samples).

The strategy implemented by the algorithms as the selection heuristic for the order of attribute tests and the maximum length (pre and pos pruning) of the tree structure are the main differences between the algorithms in the class of decision tree models. For a regression decision tree learner, the number of leaves in each tree or the granularity given by the model is also an important parameter, as it gives the number of categories that the continuous output is being discretized.

Other categories of decision tree algorithms include also the creation and use of ensembles of trees that use multiple strategies to create a varied set of trees that later will vote or compete for the best output for a given instance. For instance, the bagging technique produces replicate training sets by sampling with replacement from the training instances and, the boosting technique uses all instances at each replication, but maintain a weight for each instance in the training set (intended to reflect that vector's importance). In either case, the multiple models produced are combined by voting to form composite model. In bagging the voting is equivalent in strength between all models while in boosting the votes are weighted by the accuracy of each model (LUGER, 2002).

This class of algorithms are known by their flexibility (non-parametric models), robustness (less sensible to heavy-tailed distributions, to outliers and to irrelevant or redundant attributes), interpretability (conclusions are made by a series of simple and local decisions based on the attribute values) and efficiency (fast greedy algorithms with a divide and conquer strategy). However, decision and regression trees suffer with problems of replication (duplication of tests in different parts of a tree structure), absent values (an unknown attribute value causes an indecision process for the model), they suffer with real valued or continuous attributes (that usually need to be discretized) and are instable (few variations on the training set can produce a completely different model) (FACELI et al., 2011).

4.2 Support Vector Machines

A notion that is central to the construction of the support vector learning algorithms is the inner-product kernel between a "*support vector*" and the vector drawn from the input space. This class of machine learning algorithms make use of a set of support vectors that separates the training instances in sets by their similarity (classification problems) or a set of support vectors which will represent functions that can produce continuous outputs with minimal deviation from the training data set (FACELI et al., 2011).

The support vectors consist of a small subset of the training data extracted by the learning algorithm (HAYKIN, 1999). Fundamentally, the support vector machine (SVM) algorithms are trying to optimize such support vectors by finding the widest possible gap between instances of different sets/classes.

Additionally, SVMs can efficiently perform non-linear classification/regression using what is called the "*kernel trick*", implicitly mapping their inputs into high-dimensional feature spaces that are linear separable. Therefore, the kernels in SVM determines the shape of the decision surface and are the main hand tuned parameter of the SVM algorithms (SMOLA and VISHWANATHAN, 2008).

Kernels are measures of similarity. Broadly speaking, machine learning algorithms which rely on the dot product between instances can map all instances by a kernel function. In other words, kernels correspond to dot products in some dot product or arbitrary space There are three main advantages of using kernels (SMOLA and VISHWANATHAN, 2008):

(i) If the feature space is rich enough, then simple estimators such as hyperplanes and half-spaces may be sufficient;

(ii) Kernels allow us to construct machine learning algorithms that use a different dot product space without explicitly computing it;

(iii) And, we need not to make any assumptions about the input space of the instances other than for it to be a set.

The simplest of all kernels are defined by linear function. They are easy to compute and are a natural representation to use for vectorial data. Polynomial kernels use a function of a degree *d* to map the instances with a larger flexibility than the linear version but with a higher cost in complexity to compute. A more statistical approach also can be applied in kernels using Gaussian functions which grants the method more robustness and generality.

There many other types of kernel functions that may be applied depending on the instances mapping desired and applications, including but not restricted to convolution, string and graph kernels.

4.3 Artificial Neural Networks

An artificial neural network is a massively parallel distributed processor made up of simple processing units, which has a natural propensity for storing experimental knowledge and making it available for use. Each processing unit (neuron) takes a number of real-valued input (possibly the output of other units) weighted by their synapses' strength and produce a single real-valued output (which may become the input to many other units) (HAYKIN, 1999).

The study of these algorithms has been inspired in part by the observation of the biological learning and thinking systems that are built of very complex webs of interconnected neurons (brain). It resembles the brain in three main aspects (HAYKIN, 1999):

(i) Knowledge is acquired by the network from its environment through a learning process;

(ii) Interneuron connection strengths, known as synapse weights, are used to store the acquired knowledge;

(iii) In rough analogy, they are also built out of a densely interconnected set of simple processing units called neurons.

They provide a general, robust and practical method for learning any kind of function from examples. It may be employed for classification, regression, clustering, reinforcement and temporal learning tasks. This class of algorithms are amongst the most effective learning models currently known for several tasks in many areas (MITCHELL, 1997).

The use of neural networks offers the following useful properties and capabilities (HAYKIN, 1999):

- Nonlinearity – The neurons of an artificial network can be linear or nonlinear so, it can solve nonlinear problems;
- Input-Output Mapping – The artificial neural networks are able to learn with the help of labeled data (supervised learning), learning from sample data and creating a nonparametric model (without any prior assumptions for the data distribution);
- Adaptivity – The model generated by the network can be constantly adapted to its environment by adjusting the synapse weights to deal with new circumstances;

- Evidential Response – The model can not only find and select patterns in the data but also indicates the confidence level it has on that decision;
- Contextual Information – Knowledge is represented in the structure and activation state of the model where each unit possibly is affected by the global activity of the model;
- Fault Tolerance – Due to the distributed nature of information stored in the network, damage in a neuron unit or synapse weight will only degrades its performance not causing a catastrophic failure;
- Very Large Scale Integrated Implementability – The parallel nature of the model able its use and make the most of parallel and distributed computing units and hardware which turns it potentially fast for several tasks;
- Uniformity of Analysis and Design – Neural networks can be modularized, integrated and shared among different applications and with different architectures;
- Neurobiological Analogy – The design of an artificial neural network can benefit from new discoveries and be motivated by their biological counterpart.

There are several kinds of artificial neural networks and associated learning algorithms. How the network is structured, which kind and number of connections, layers and units, together with the procedure applied to produce the outputs in each unit by the use of distinct activation functions are the main factors that distinguish different learning algorithms and models in this class. For instance, recurrent networks (networks with feedback loops) are often associated to temporal and reinforcement learning and deep networks (networks with a huge number of layers) are associated to image/video learning tasks.

Moreover, these meta-parameters of the model are a mixture of pre-defined parameters that must to be tuned by hand (some are defined by the kind of task being solved) with parameters that are automatically optimized by one of a vast set of learning algorithms.

These learning algorithms can be categorized in five classes (HAYKIN, 1999):

(i) Error-correction learning and their algorithms such as the backpropagation use gradient descent to tune the network parameters to best fit a training set of input-output pairs of values. The computation of such gradients may be so complex due to the network structure and problem size that several heuristics and different methodologies were developed by many authors as alternative solutions;

(ii) Memory-based learning store past experiences explicitly in a large memory of correctly classified input-output examples;

(iii) Hebbian learning use as basis the associative learning where the synchrony between neurons strengthen their synapses and asynchrony weakens of eliminates synapses;

(iv) Competitive learning implies that the output neurons compete among themselves to become active meaning that only one output neuron can be active at the same time;

(v) Boltzman learning is rooted in statistical mechanics and it is a stochastic learning algorithm. Artificial neural networks that apply this kind of learning are called Boltzman machines.

Hebbian and competitive learning are self-organized, unsupervised learning and motivated by neurobiological considerations. While the other methods are supervised learning which empowers them to fulfill better tasks such as classification and regression tasks.

The techniques presented in this section were applied, analyzed and compared in ours experiments to approximate the centrality measures together with other approximation methodologies not based on machine learning. These experimentations will be detailed and discussed in the next section.

It is important to mention that in my experiments I do not applied Recurrent Neural Networks (RNNs) and Long-short Term Memory Networks (LSTMs) because the training data does not have any temporal dependency between attributes and instances (main characteristics dealt by this kind of ANNs). Convolutional Neural Networks (CVNs) were also not considered in the experiments because the main idea in our methodology is to deal with the simplest inputs as possible and not deal with the entire graph representation, which would be the case with the use of CVNs.

Considering that ANNs and SVMs are known to be equivalent in theory (HAYKIN, 1999), considering their capabilities, I choose to optimize the one that I feel more comfortable with, the ANNs, at the same time that I did also a performance (accuracy) comparison in the approximation of centrality measures between both techniques to check if one has a clear advantage over the other or not.

5 APPROXIMATION TECHNIQUES FOR CENTRALITY MEASURES

Typical centrality measures algorithms do not scale up to graphs with billions of edges (such as large social networks and Web graphs). Even though their algorithms are polynomial in time, their computation requires days or even months for massive networks and are inefficient to use in real-time applications (BADER et al., 2007; EPPSTEIN and WANG, 2004). Closeness metric, for instance, takes about 5 years to compute for a network with 24 million vertices and 29 million edges (COHEN et al., 2014).

This can be even more critical when one needs to compute many centrality queries, particularly when one is interested in the centrality of all vertices or whenever the network structure is dynamic through the time. This feature is common to most real networks, which are constantly changing and evolving in time (COSTA et la., 2008; EASLEY and KLEINBERG, 2010).

In this section, I describe the most common method for sampling approximation techniques and then detail the methodology based on machine learning techniques for network centrality learning and computation.

5.1 Vertices Sampling Techniques

The first technique proposed in the literature to approximate the centrality measures are based on vertices sampling (EPPSTEIN AND WANG, 2004). The underlying premises behind a sampling technique is simple: one should compute the exact centrality value for a predefined number of sampled vertices and then estimate the centrality value of the others based on such computation. For instance, the betweenness and closeness centralities share a quite similar foundation for the sampling technique:

- One computes the single source shortest path (SSSP) for a given number of sample vertices;
- Each SSSP tree gives the exact centrality value for its source vertex;
- At the same time, one can use them as an approximation for the other vertices considering that all previously computed SSSP trees are partial solutions for such vertices;
- Therefore, a given vertex not sampled will have its centrality value approximated by an average result given by all SSSP trees from the sampled vertices.

An algorithm for such objectives was defined and tested in real case scenarios by Bader et al. (2007) for betweenness and by Eppstein and Wang (2004) for closeness centrality. However, the simple approach given by the sampling technique has a few drawbacks and leads to relevant questioning such as:

(i) How many vertices should be sampled and how should one select them?

(ii) Or, how can one efficiently parallelize the sampling technique considering that it is not possible any longer to compute each vertex centrality independently?

Brandes and Pich (2007) studied the vertices selection problem. They proposed several heuristics to choose vertices for the sampling techniques, starting with simple ones, such as picking the high degree vertices and finishing with more complex ones, which considers vertices distances and mixed strategies. They concluded that picking vertices uniformly at random on average is the best strategy when considering different types of network structures.

The second question still is an open research question. To provide an answer for such question or at least measure the effectiveness of the sampling technique with and without parallelism I built an experimental setup that will be explained and discussed in detailed in the next subsection.

5.2 Building Sampling Experiments

In order to make the concepts clearer to the reader, I put the ideas into a practical experimentation. This will serve to illustrate the capabilities of the method in validation and comparative analyses.

I shall illustrate the effect of sample sizes in a parallelized version of the sampling algorithms in real case scenarios using 30 real-world networks from four freely available data repositories. The selected networks are symmetric and binary (unweighted edges) with sizes ranging from a thousand to near two million vertices. Only the largest connected component (LCC) was used for the tutorial experiments, which was computed for all analyzed networks. The list of all real networks with their respective size and the size of its LCC (proportion w.r.t. the number of vertices), classified by network type and grouped by data repository is presented in Table 5.1.

In the sequel, I computed each of the four exact centrality measures (eigenvector – CE, betweenness – CB, closeness – CC, and degree – CD) for all the thirty real networks depicted in Table 5.1. The computation of eigenvector and degree centralities is sequential, while

betweenness and closeness centralities are computed simultaneously with a merged algorithm keeping the same time complexity upper bounds and using parallelism (BRANDES, 2008).

It is important to notice that the centrality value for each vertex of a network can be computed fully independent from the others in a parallel environment just at the expense of a higher memory cost. All algorithms were programmed in C and the parallel computations used the native OpenMP (Open Multi-Processing interface). The computation of the centrality measures was realized with the help of a computer cluster at the supercomputing national center in the federal university of Rio Grande do Sul (CESUP/UFRGS). The machine configuration used for such task was an SGI Altix Blade with 2 AMD Opteron dodeca-core (i.e. 24 real core processing units in total) with 2.3GHz, 128KB L1 Cache and 512KB L2 Cache per core, 12MB L3 cache total and 64GB DDR3 1333MHz RAM memory, Red Hat Enterprise Linux Server 5.4.

The computation time for the centrality measures used in our experiments is presented in Table 5.2, and represents the actual time spent in the execution of the exact algorithms for each of the metrics. I omitted from the table the degree centrality because it required less than 1 second to compute for all the networks. Table 5.2 also presents the mean time required for the computation of a sample-based approximation algorithm for betweenness and closeness centralities. The computation of the approximation algorithm was run in the same machine, but in a sequential environment.

I tested also the parallelization of the approximation algorithms but it demonstrated to be highly inefficient because it required a large number of dependent variables between different threads and a larger amount of memory. Parallelization reduced the execution time in about half but used 24 cores and 24 times more memory in the parallel environment.

I adopted two sample sizes for each network: 2.5% and 5% of the number of total vertices for the approximation algorithms. The samples were uniformly randomized, following the work of Brandes and Pich (2007) that found out that this is the optimal strategy for such task. Five independent trials with distinct random seeds for each sample size for each network were executed in my experiments.

TABLE 5.1 – Experimental Data: Real Netwroks Description

Network	Type	Vertices	Edges	LCC %
Stanford Large Network Dataset Collection				
Autonomous Systems AS-733 (LESKOVEC et al., 2005)	Routers	6,474	12,572	100.00
Oregon-1 (LESKOVEC et al., 2005)	Routers	11,174	23,409	100.00
Oregon-2 (LESKOVEC et al., 2005)	Routers	11,461	32,730	100.00
Astro Physics (LESKOVEC et al., 2007)	Collaboration	18,772	196,972	95.37
Condensed Matter (LESKOVEC et al., 2007)	Collaboration	23,133	91,286	92.35
General Relativity (LESKOVEC et al., 2007)	Collaboration	5,242	13,422	79.32
High Energy Physics (LESKOVEC et al., 2007)	Collaboration	12,008	117,619	93.30
High Energy Physics Theory (LESKOVEC et al., 2007)	Collaboration	9,877	24,806	87.46
Amazon (YANG and LESKOVEC, 2012)	Co-Purchasing	334,863	925,872	100.00
DBLP (YANG and LESKOVEC, 2012)	Collaboration	317,080	1,049,866	100.00
Youtube (YANG and LESKOVEC, 2012)	Social	1,134,890	2,987,624	100.00
Brightkite (CHO et al., 2011)	Social	58,228	212,945	97.44
Gowalla (CHO et al., 2011)	Social	196,591	950,327	100.00
Enron (KLIMMT and YANG, 2004; LESKOVEC et al., 2009)	Email	36,692	180,811	91.83
Texas (LESKOVEC et al., 2009)	Road	1,921,660	1,879,202	70.31
Facebook (MCAULEY and LESKOVEC, 2012)	Social	4,039	82,143	100.00
Social Computing Data Repository				
Blog Catalog 3 (ZAFARANI and LIU, 2009)	Social	10,312	333,983	100.00
Buzznet (ZAFARANI and LIU, 2009)	Social	101,168	2,763,066	100.00
Delicious (ZAFARANI and LIU, 2009)	Social	536,108	1,365,961	100.00
Douban (ZAFARANI and LIU, 2009)	Social	154,907	327,162	100.00
Foursquare (ZAFARANI and LIU, 2009)	Social	639,014	3,214,986	100.00
Hyves (ZAFARANI and LIU, 2009)	Social	1,402,611	2,184,244	100.00
Livemocha (ZAFARANI and LIU, 2009)	Social	104,438	2,193,083	99.68
BGU Social Networks Security Research Group				
The Marker Café (FIRE et al., 2011; FIRE et al., 2013)	Social	69,411	1,644,781	99.86
The Koblenz Network Collection				
US Power Grid (WATTS and STROGRATZ, 1998; KONECT, 2016)	Supply Lines	4,941	6,594	100.00
Catster (KONECT 2016)	Social	149,700	5,447,465	99.42
Dogster (KONECT 2016)	Social	426,820	8,543,322	99.92
Hamster (KONECT 2016)	Social	2,426	16,098	82.44
Euroroad (KONECT 2016; ŠUBELJ and BAJEC, 2011)	Road	1,174	1,305	88.50
Pretty Good Privacy (BOGUÑÁ, 2004; KONECT, 2016)	Communication	10,680	24,316	100.00

TABLE 5.2 – Metrics Computation time with the Real Networks

Network	Time (hh:mm:ss)			
	Exact Algorithm		Sample 5%	Sample 2.5%
	C_E	C_B and C_C	C_B and C_C	C_B and C_C
Euroroad	< 1s	< 1s	00:00:52	00:00:23
Hamster	< 1s	00:00:03	00:01:42	00:00:49
Facebook	< 1s	00:00:05	00:03:31	00:01:42
General Relativity	< 1s	00:00:05	00:03:29	00:01:43
US Power Grid	< 1s	00:00:05	00:04:13	00:02:02
Autonomous Systems AS-733	< 1s	00:00:11	00:05:35	00:02:45
High Energy Physics Theory	< 1s	00:00:25	00:07:21	00:03:36
Pretty Good Privacy	< 1s	00:00:25	00:09:07	00:04:33
Oregon-1	< 1s	00:00:36	00:09:32	00:04:46
Oregon-2	< 1s	00:00:37	00:09:50	00:04:49
High Energy Physics	< 1s	00:01:25	00:09:42	00:04:48
Condensed Matter	< 1s	00:03:20	00:18:54	00:09:23
Blog Catalog 3	00:00:04	00:03:57	00:09:42	00:04:48
Astro Physics	00:00:01	00:04:42	00:16:19	00:08:00
Enron	00:00:01	00:10:28	00:30:46	00:15:12
Brightkite	00:00:02	00:25:46	00:53:41	00:26:26
Douban	00:00:03	02:10:51	02:45:48	01:23:02
The Marker Cafe	00:00:29	03:05:17	01:50:32	00:54:31
DBLP	00:00:09	03:33:41	07:40:50	03:49:58
Buzznet	00:00:47	03:41:01	03:10:24	01:35:47
Amazon	00:00:16	04:19:25	08:45:19	04:23:10
Livemocha	00:00:39	04:30:08	03:19:42	01:40:25
Gowalla	00:00:10	04:32:46	04:31:08	02:13:08
Catster	00:01:17	06:16:43	07:04:34	03:15:08
Delicious	00:00:20	09:47:57	15:34:38	07:53:01
Foursquare	00:00:44	24:23:38	32:33:36	16:04:30
Texas	00:00:10	33:38:20	49:06:11	24:07:02
Dogster	00:01:39	120:30:45	27:05:10	17:11:20
Youtube	00:01:03	188:07:33	44:04:45	22:28:10
Hyves	00:00:41	213:16:27	54:08:52	26:24:56

Notice that the times presented in Table 5.2 comprise the computation of both metrics (betweenness and closeness) summed up. Besides, the computation of both algorithms was simultaneous taking advantage of the fact that they share similar parts and so the performance can be increased.

I also tested a sequential version for the betweenness and closeness centralities in the smaller networks (up to a hundred thousand vertices) used in the experimentations. They took around 16 times more computation time than their parallel versions, which indicates that the parallel version of the algorithm granted only about 2/3 of gain despite all threads being fully

independent and may be related to the fact that the physical configuration of the processor units in the computer are not fully independent.

Notice that eigenvector and degree centralities are feasible even for massive networks, requiring orders of time less to compute than betweenness and closeness centralities, even though the computation of the previous were sequential and the computation of the latter made use of 24 cores in a parallel environment.

One can also observe the overhead effect of the sampling algorithms. Such algorithms maintain the same asymptotic time complexity upper bound, but at the expense of higher constant variables. For such reason, they are only viable when the size of networks compensates this effect, which, in the tests, occurred only for the networks with more than one million vertices when compared with the parallelized version of the exact algorithms.

I compared the results of the approximation algorithms (sample sizes of 2.5% and 5.0%) with the exact metrics using the Kendall τ-b correlation coefficient, which is a nonparametric measure of strength and association (interval [-1,1]) that exists between two variables measured on at least an ordinal scale. The 5% sample achieved a correlation of 0.9548, while the 2.5% sample achieved 0.9461. By such means the results show that the larger sample size, which required twice the time to compute in average, did not compensate its cost by showing a just a small increase in the algorithm accuracy at approximating the centrality measures.

Therefore, I showed, by an experimental analysis, that one needs just a small fraction of vertices to approximate the centrality of all vertices of a huge real network with high accuracy, fact that is not empirically tested in the literature. However, at the same time, I confirmed that the parallel version of the approximation centralities was not efficient and so the sampling algorithms only started to pay off in networks of considerable size (more than a thousand vertices in my experiments).

Next, I explain a faster (linear bounded) approximation technique for centrality measures that uses machine learning methods.

5.3 Machine Learning Methods

I propose and develop in my work a methodology based on machine learning to approximate the centrality measures rank (a regression task) but the use of such techniques for such problem is not novel in the literature. KUMAR et al., 2015 have tried and experimented with approximation methodologies based mainly on machine learning and artificial neural

artificial networks. However, his work is limited because he only focused the application of the technique to the approximation of eigenvector and PageRank centrality measures, which exact computations are feasible even for massive networks (Costa et al., 2008). In addition, he did not try to optimize the neural network meta-parameters and structure, using a generic neural network model instead, he did not provide a generalizable methodology for to apply the same technique to other centrality measures and how one can get enough training data.

Here, I propose and explain a neural model methodology that has two major strengths:

(i) Its adaptability – they can be used to approximate several distinct centrality measures;

(ii) And its efficient computation time – they are able to compute the centrality measures a lot quicker than any other method and the methodology comprises also a fast and easy learning model.

Moreover, I provide a straightforward methodology in which a complex network model is used to properly generate synthetic networks to offer plentiful training data for the model. I also optimize the model considering several combinations of parameters and test it on real world networks comprising several thousands of vertices (GRANDO and LAMB, 2015; 2016; 2018).

The methodology underlying the machine learning method developed and tested in my research is divided into four main steps.

(i) First, the training data is acquired by using a complex network model to generate synthetic networks in which the centrality measures are computed. This is the data used for training the artificial neural network;

(ii) Second, the training algorithm, network size and meta-parameters are selected for training the artificial neural network;

(iii) Third, the accuracy (over test set) of the model generated with the artificial neural network is compared with models generated by other machine learning techniques and also other approximation techniques based on sampling methodologies;

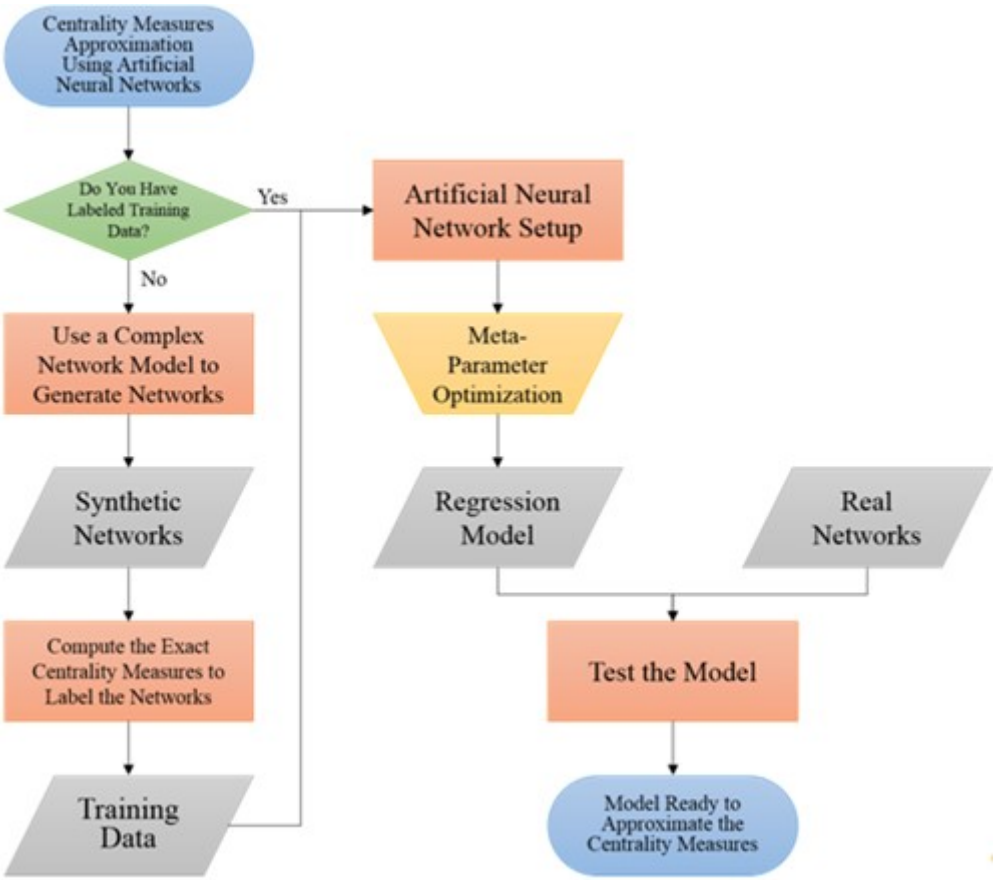
(iv) Finally, the regression model is applied to real-world networks in an application and its accuracy is measured with the computation of the exact centrality measures.

These steps are illustrated in a flowchart to facilitate the methodology understanding and reproduction of experiments using the techniques described here. Each one of these four steps will be detailed in the following subsections.

Figure 5.1 summarizes how one can generate (set up and train) a model for regression tasks in several applications using the same methodology proposed and applied in my

experimentations. Each shape that it is present in the flowchart shown in figure 5.1 have a different conceptual meaning. The ellipses (in blue) in the figure stands for the flowchart beginning and ending, while the lozenges (in green) are decision steps, the rectangles (in red) are actions/procedures, the rhomboids (in grey) are inputs and outputs of the procedures and the trapezium (in orange) are loop/repeated procedures.

Figure 5.1 – Summary of the Machine Learning-based method for centrality measures



5.3.1 Training Data Acquisition

When using any supervised machine learning method/algorithms, one important step is to obtain enough and consistent training data and to select the appropriate input data for the model. Since obtaining data from real networks is a costly process and that the computation of the centrality measures in huge networks is very expensive, one can generate his own networks for such purpose using a complex network model.

The complex network model enables one to generate many synthetic networks with the properties enjoyed by real networks. Moreover, it allows one to generate smaller networks that

reduce the computing costs to calculate the exact centrality measures, but – more importantly - keeping the most relevant structural properties presented by massive real-world networks.

The BTER complex network model was used to obtain enough and consistent training data. The BTER complex network model was chosen for such a task as it is capable to reproduce most real-world networks structural properties. Moreover, it is easy to implement and configure and capable to generate plentiful networks with reduced size keeping the most relevant structural properties presented by massive networks.

BTER requires two configuration parameters: the desired degree distribution and the clustering coefficient (which can be configured as a global value, by vertices degree or by community structure). More information about this complex network model is presented in subsection 2.7.

In my experiments, I applied both a heavy-tailed and a lognormal distribution as degree distribution to provide generic data to train the model. Both distributions are known as the best representatives of most real networks' degree distribution studied in the literature (BOCCALETTI et al., 2006; MITZENMACHER, 2003). I selected as desired maximum clustering coefficients for the BTER model random values in the real numbers' interval [0.3, 0.7] as they are well representatives for most real networks (COSTA et al., 2008).

Section 2.7 present further details about the generation of these networks with the BTER model and presents the Table 2.1, that summarizes and explains in detail the formula of each function used to model the degree distribution and the parameter values used in the experiments.

The training data contained 10 networks for each kind of degree distribution (Table 2.1) with sizes ranging from 100 to 1,000 vertices, totaling 600 synthetic networks and 330 thousand vertices. These relatively small sizes were selected to enable the computation of the exact ranks of the metrics (betweenness and closeness) used as labels during the training (supervised learning).

I also generated networks based on the degree distributions and clustering coefficients of six real networks as they presented the worst (Amazon, Euroroad, Facebook and PowerGrid networks) and the best (Blog3 and Foursquare networks) overall results amongst the tests obtained by the model trained with the generic training data amongst the thirty real networks described in Table 5.1.

A transformation function (parameters estimated for each degree distribution using as reference a heavy-tailed function model) was applied in the degree distribution to reduce the size of the generated networks. In other words, for each of the networks was estimated the parameter of the heavy-tailed function that best represent its degree distribution slope and that

parameter than is used to retain the same slope for smaller networks (it can be checked visually in Figures 5.2 through 5.7).

A total of 3600 specific networks were generated with the BTER for this experimental setup (600 for each set of parameters), with sizes of one thousand, two thousand and three thousand vertices.

The synthetic networks with size one thousand were not used to train the artificial neural network models because I have considered enough the training samples obtained with the other synthetic network sizes. The networks with larger sizes also behave more likely to the real ones capturing more detailed structures. Consequently only 2400 specific networks were applied in the training and experimentations. These networks were used to compare the effect of specialized training data (based on the parameters of a given group of real networks domain) versus generic data (based on generic degree distribution functions as seen in Table 2.1).

Figures 5.2 to 5.7 present the original degree distribution (logarithmic scale in both axis) of each real network and the generated degree distribution by the BTER model for the networks with sizes ranging from one to three thousand vertices.

The Euroroad network is already a small sized network with just 1174 vertices (Table 5.1), therefore I configured the BTER model to generate synthetic networks with the exact same size of the original one.

Figure 5.2 – Amazon Degree Distributions

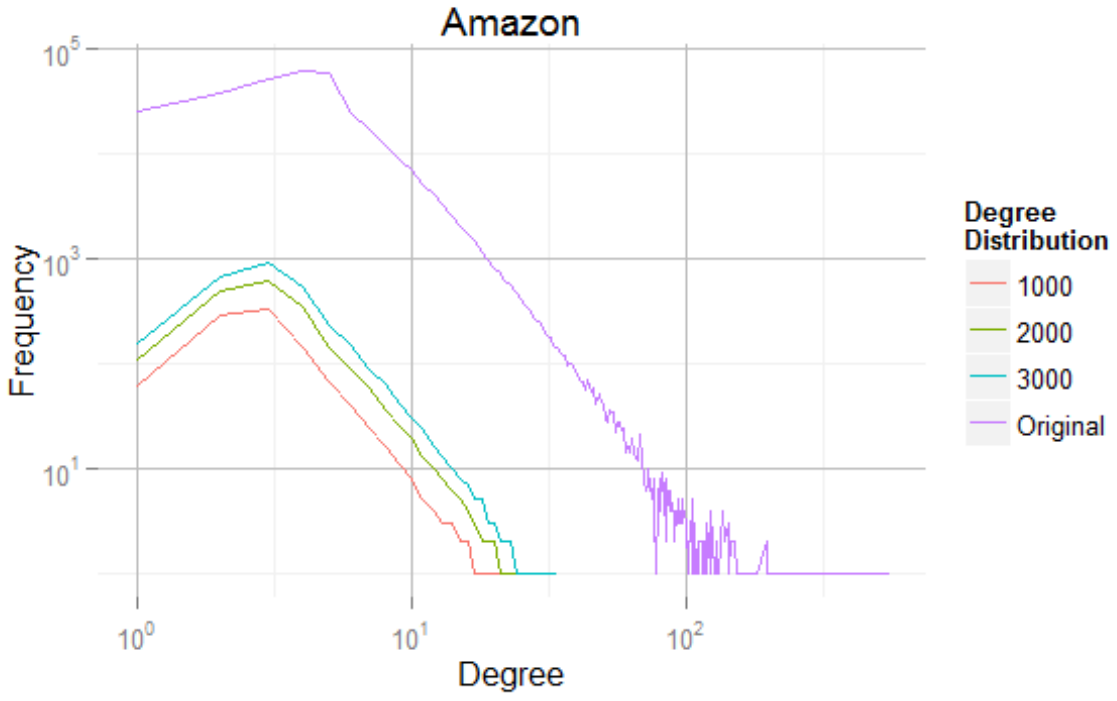


Figure 5.3 – Blog3 Degree Distributions

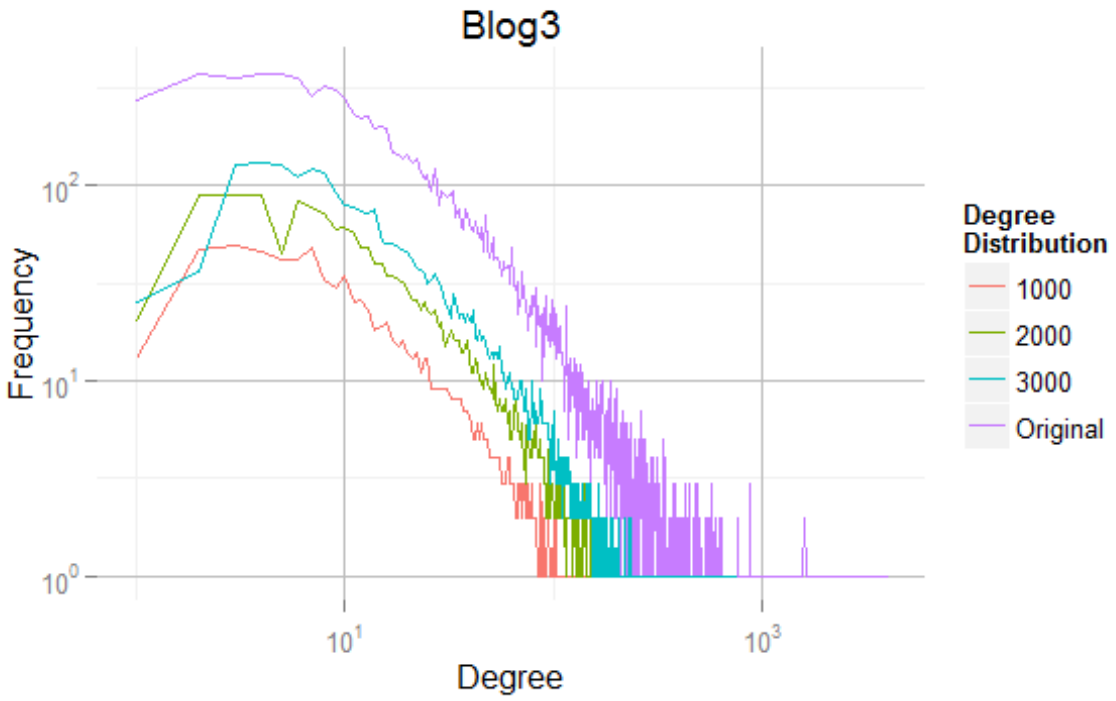


Figure 5.4 – Euroroad Degree Distributions

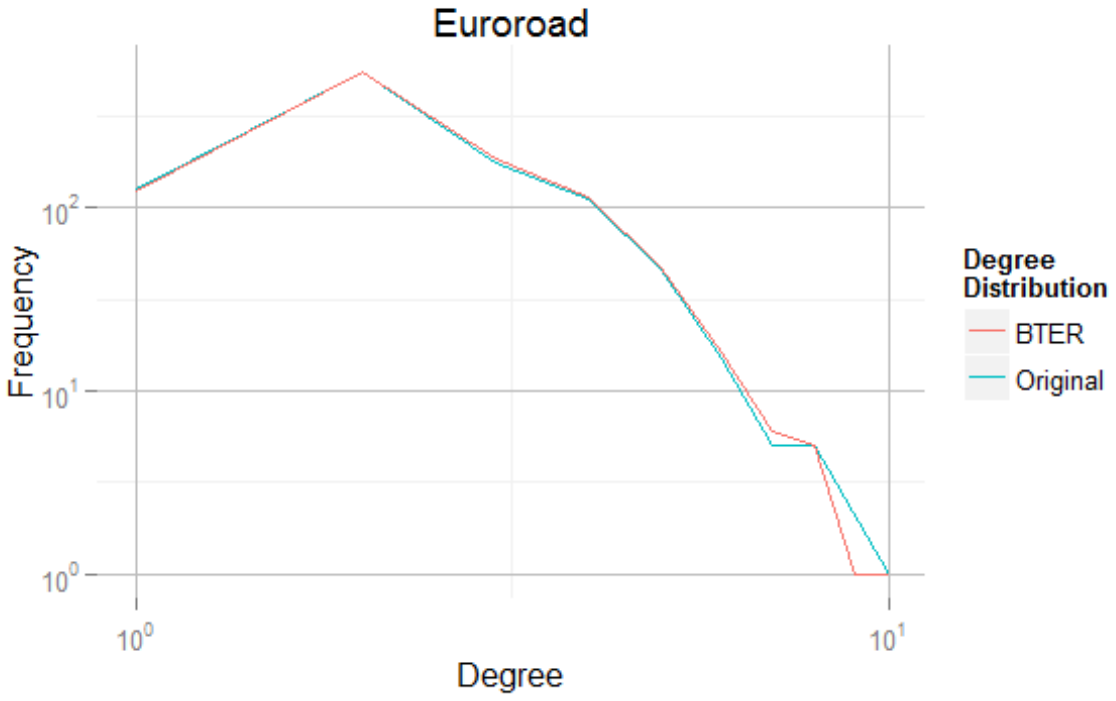


Figure 5.5 – Facebook Degree Distributions

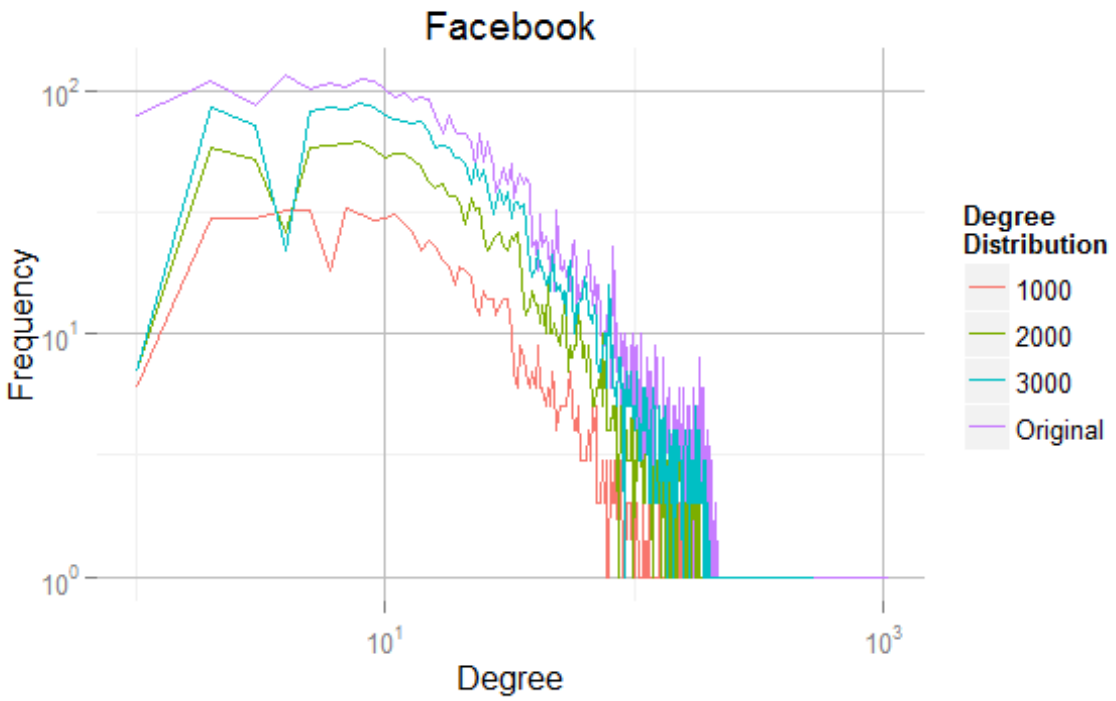


Figure 5.6 – Foursquare Degree Distributions

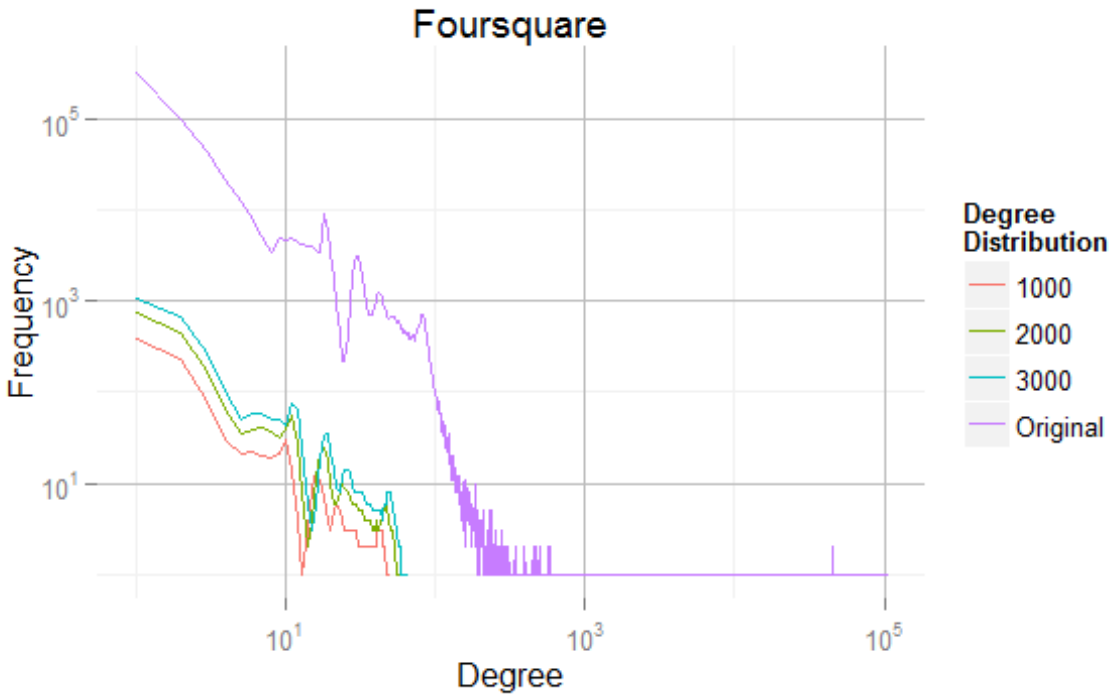
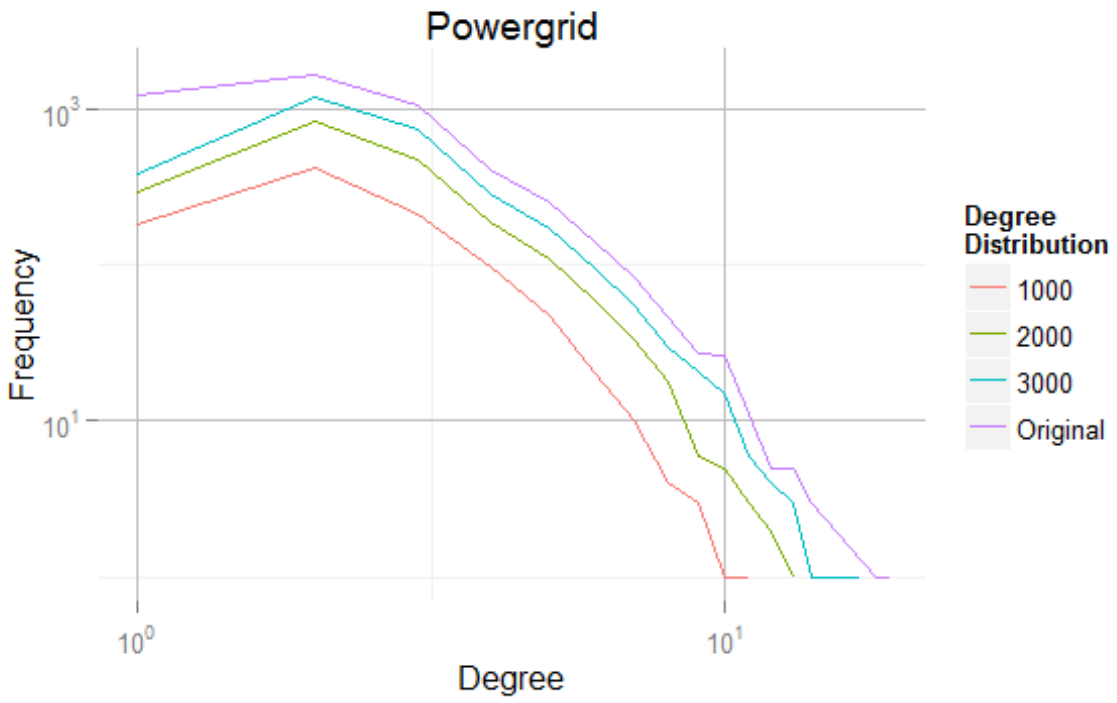


Figure 5.7 – Powergrid Degree Distributions



It is noticeable that the BTER model is capable to match really close the degree distribution of all real networks (despite their very distinct size range) at the same time it generates networks with diminished size.

Moreover, the clustering coefficients of the synthetic generated networks were very similar to the real networks providing just a margin of 10% above or below of the desired value for all networks tested. However, the optimal clustering parameter was not as simple as the degree distribution to configure in the BTER model to achieve such similarity because the parameter needed to be configured accordingly to the new size network (much smaller than the originals) and usually is a larger number than the desired final clustering (due to restrictions of the BTER model – explained in detail in subsection 2.7). Therefore, the optimization process applied to find the clustering parameter was mainly based on binary search.

The second issue one has to deal when training a machine learning model is the selection of the proper input attributes for the problem. Each centrality measure uses complex structural properties of the graph representing the network and the computation of most of these properties are the main reason that centrality measures are time expensive. For such reason, and considering the fact that simpler centralities are quite correlated to the other centralities (check subsection 3.5, I selected the fastest centrality measures (degree and eigenvector), which are computationally feasible even for massive networks and are highly related to the other metrics

(Table 3.2), as the necessary input attributes for the machine learning model in my experiments. This provides generality to this technique as such metrics summarize most information needed to compute the other extant and more complex metrics. Thus, I compute degree and eigenvector to serve as input data and the exact values of betweenness and closeness as desired output/label values (supervised learning) in the networks generated with the BTER model.

Closeness and betweenness centralities will be the target values in our experiments but the same technique may be applied for any other centrality measure. These two metrics were chosen because they are the most applied centrality measures in the literature in several applications. At the same time, their exact algorithms computation is unfeasible or require too much time to compute in massive networks and dynamic networks. The increasing availability and interest of study on such huge and dynamic networks created an urge for approximation techniques for both measures, as they are important analysis tools for many areas (DAS et al., 2018).

I computed the rank of each vertex in each centrality measure, as we are interested in the approximation of the order of such vertices and not in the absolute centrality values. These will help to improve the performance of the model without restricting its applications in real world environments because the rank is usually preferable to deal in the application of centrality measures than the absolute value is in most environments (COSTA et al., 2008). Consequently, a sample input of training data will contain the rank a vertex considering the eigenvector and degree centralities (2 inputs) and the outputs/label/desired values will be the rank of the same vertex in one of the other centralities (betweenness and closeness in my experiments).

It is important to highlight that when one is considering the use of the same technique to approximate other centrality measures (e.g. walk betweenness, hierarchical closeness or subgraph centralities), the input information can remain the same, the simplest centrality measures (degree and eigenvector). Any desired measure needs to be computed in the synthetic networks to serve as output labels during the training. Notice that such computation is feasible for any target centrality measure due to the use of the BTER complex network model that generates smaller synthetic networks with properties similar to that of the massive networks that will later be used in the application environment.

Both input and desired values are first normalized by the size of the network, then to belong to the interval [-1, 1] and finally to have zero mean and variance equal to one. The preprocessing helps the configuration of the model and speeds up the training process for the machine learning algorithm.

The first machine learning algorithm tested in my experiments is the artificial neural networks due to their ability to be general purposes method and a proper regression methodology for many tasks in the literature.

The next subsection will detail the configuration and training of such artificial neural networks while I try to determine the optimal setup for the task in hand. Later, the performance of the models generated with such methodology will be compared and tested against the performance of other common machine learning techniques that are frequently used in the literature to generate regression models.

5.3.2 Artificial Neural Network Training

The appropriate training of an artificial neural network can be a complex issue. Many techniques and methodologies, algorithms and tools can be applied in specific tasks. For such reason, I applied the neural network toolbox from MATLAB 2015 to configure and train the neural networks. This toolbox comprises most of the algorithms that are well established in the field with a built-in parallelism pool and a robust implementation for industrial, academic and research use.

Before one starts the training, one needs to select several meta-parameters for an artificial (neural) network, such as size and structure of the network, learning algorithm, and learning parameters. Our objective here was to find out the most efficient configuration of parameters considering both its computational costs and solution quality. Therefore, in the experiments, I initially optimized these parameters using 10-fold cross-validation (over the generic data described in Section 5.3.1) to find out the best configuration for these meta-parameters to approximate the centrality measures.

I applied the batch method which is typically used in the literature to update the weights of an artificial neural network because it is more robust to outliers (MITCHELL, 1997). In batch training, the adjustment delta values are accumulated over all training items to give an aggregate set of deltas and then they are applied to each weight and bias.

The quality of the solution was measured by the determination coefficient (R^2) using only the test set results, which considers 10% of total data.

I have selected the fully-connected feedforward multilayer perceptron neural network architecture (with bias, validation stopping criteria, mean square error as error function and tangent hyperbolic as activation function) as first approach as this model is simple and also

robust to outliers, generates a highly flexible model and it is easy to implement and train using the backpropagation learning algorithm.

First, I experimented with several network's sizes (numbers of neurons and hidden layers). The number of neurons of the networks with more than one hidden layer was selected in a way that they match the total number of learning parameters of the network with only one hidden layer. In this way, one can make a direct comparison of the effect caused by the number of layers and discover the most efficient configuration of parameters considering that they have a close number of parameters. Table 5.3 shows all the configuration of network structures tested, i.e. the number of layers, the number of neurons in each layer and the total number of parameters.

TABLE 5.3 – Artificial Neural Networks Structure Sizes

Number of Neurons in Each Hidden Layer (Total Number of Parameters)	**One Hidden Layer**	**Two Hidden Layer**	**Three Hidden Layer**
	7 (21)	3 (21)	2 (18)
	11 (33)	4 (32)	3 (33)
	25 (75)	7 (77)	5 (75)
	56 (168)	11 (165)	8 (168)
	84 (252)	14 (252)	10 (250)
	175 (525)	21 (525)	15 (525)
	299 (897)	28 (896)	20 (900)
	532 (1596)	38 (1596)	27 (1593)

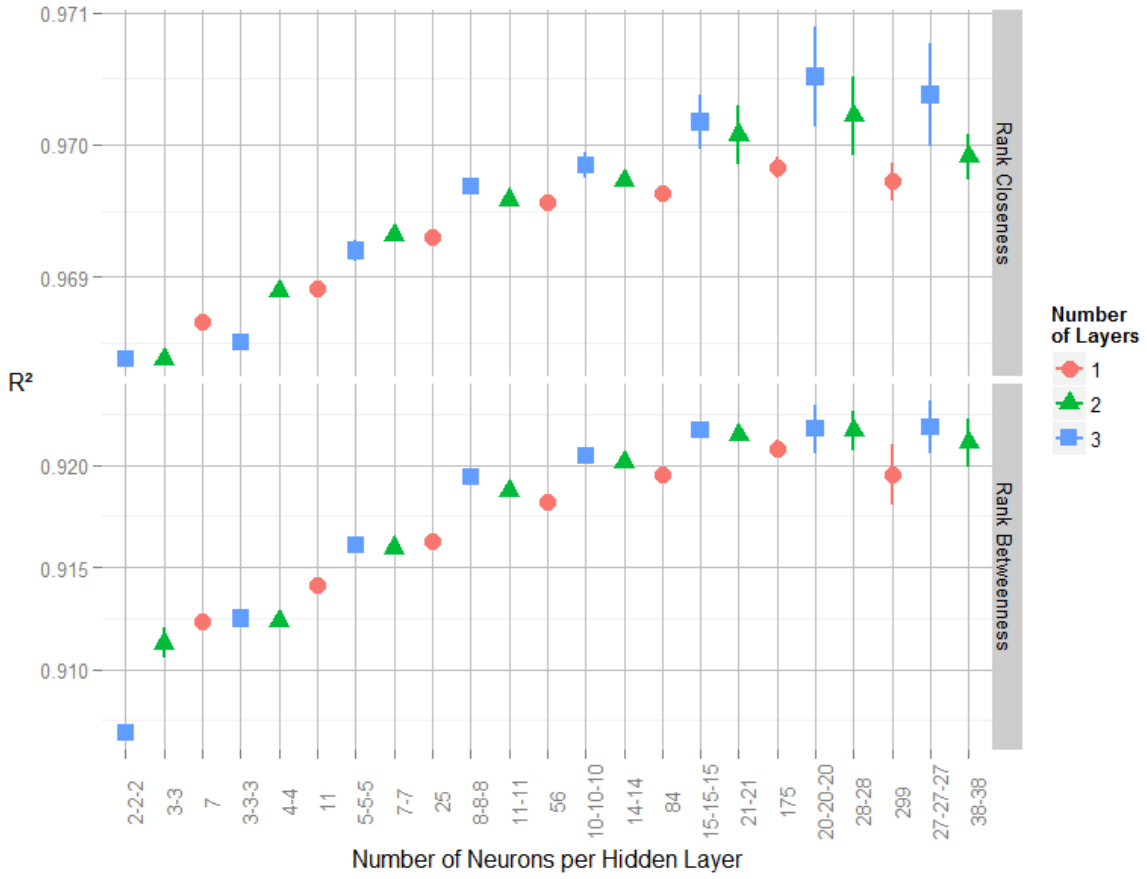
Notice that the more parameters the network comprises, the better is its performance in the training set, up to a tipping point where the complexity of its training computation does not pay off and training efficiency decays. It is important to take in consideration the fact that a large number of parameters may not result in a better performance at production phase due to the poor generalization of the model. Therefore, an important objective is also to identify such tipping point in the number of parameters to optimize the learning and model's performance.

I selected the training algorithm suggested by MATLAB environment as default (the Levenberg-Marquardt algorithm) to train the neural networks in this stage. Additionally, I set the activation function of all hidden layers to hyperbolic tangent and the activation function of the output layer as a linear function (due to the regression task in hand).

Figure 5.8 present the summary of results for each centrality measure ranked (10 trials) with 99% confidence intervals (i.e. the expected mean value is within the shown interval with

99% chance if the experiment is repeated). When the interval does not appear in the figure, it means that the variance is too small to be significant in this scale.

Figure 5.8 – Outline of Results for Different Artificial Neural Networks Structures



The results of the experiments show that a three-hidden layer network with 20 neurons in each layer was the architecture that presented the best solution quality.

In the next stage of training, I experimented with all different kinds of optimizers for the backpropagation algorithms available in MATLAB's neural network toolbox for feedforward networks (notice that, of course, many freely available machine learning software tools offer several versions of such algorithms). I tested all the combinations of previously selected network architectures with all the available training algorithms.

Table 5.4 depicts the training algorithms applied to artificial neural networks and their respective selected parameter values for this experimental stage.

The preliminary parameters are the ones selected by default by the MATLAB environment. Figure 5.9 presents the performance of the different algorithms.

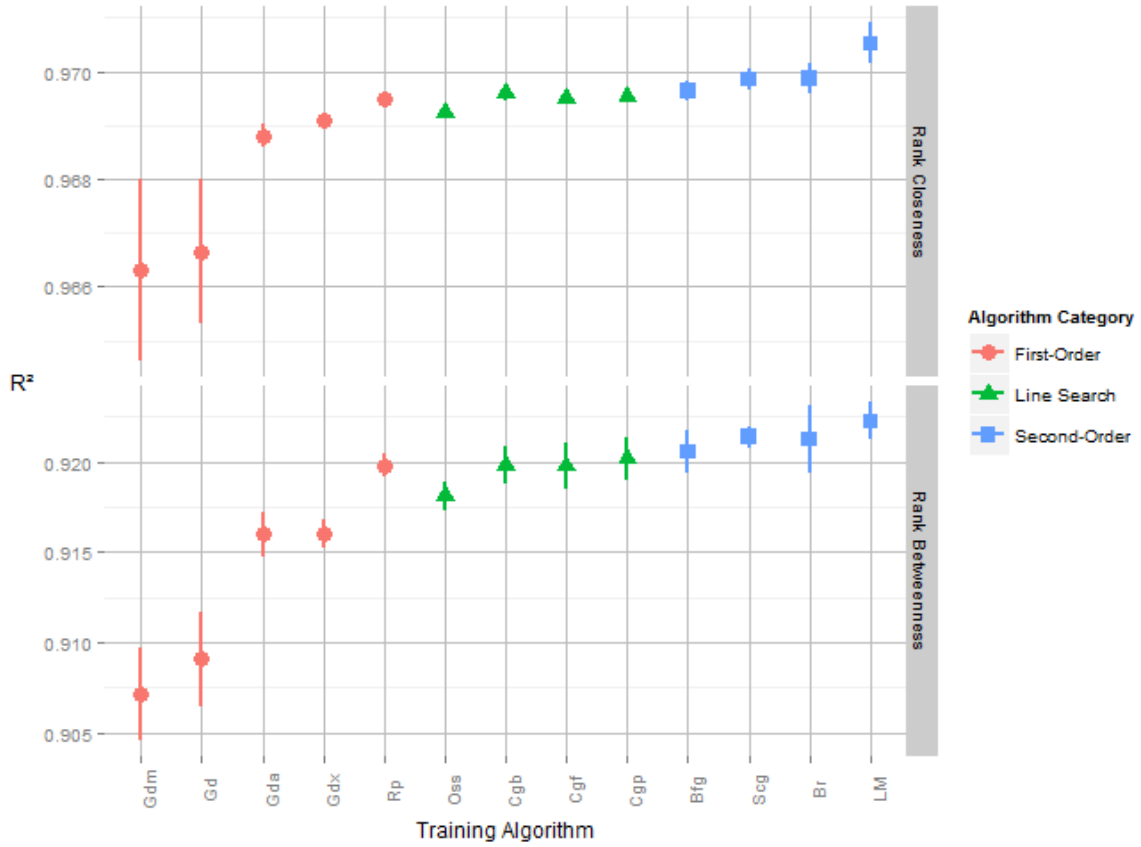
TABLE 5.4 – Parameters of the Backpropagation Algorithms

Algorithm	**Parameters**
Gradient Descent (Gd) (HAGAN et la., 1996)	Learning rate = 0.01
Gradient Descent with Momentum (Gdm) (HAGAN et la., 1996)	Learning rate = 0.01 Momentum constant = 0.9
Variable Learning Rate Gradient Descent (Gdx) (HAGAN et la., 1996)	Learning rate = 0.01 Momentum constant = 0.9 Increase/decrease ratio to learning rate = 1.05/0.7
One-Step Secant (Oss) (BATTITI, 1992)	Linear search = 1-D minimization backtracking (DENNIS and SCHNABEL, 1983) Initial step size = 0.01 Scale factor to sufficient performance reduction = 0.001 Scale factor that determine sufficiently large step size = 0.1 Step size lower/upper limit = 0.1/0.5 Minimum/maximum step length = 10^{-6}/100 Linear search tolerance = 0.0005
BFGS Quasi-Newton (Bfg) (GILL et la., 1981)	The same parameters and values as the method above.
Polak-Ribiére Conjugate Gradient (Cgp) (SCALES, 1985)	Linear search = 1-D minimization using Charalambous' method (CHARALAMBOUS, 1992) Initial step size = 0.01 Scale factor to sufficient performance reduction = 0.001 Scale factor that determine sufficiently large step size = 0.1 Scale factor to avoid small performance reductions = 0.1 Linear search tolerance = 0.0005
Fletcher-Powell Conjugate Gradient (Cgf) (SCALES , 1985)	The same parameters and values as the method above.
Conjugate Gradient with Powel/Beale Restarts (Cgb) (POWELL, 1977)	The same parameters and values as the method above.
Scaled Conjugate Gradient (Scg) (MØLLER, 1993)	Change in weight for the second derivative approximation = 5.10^{-5} Regulation of the Hessian indefiniteness = 5.10^{-7}
Resilient Backpropagation (Rp) (RIEDMILLER and BRAUN, 1993)	Learning rate = 0.01 Initial weight change = 0.07 Increment/decrement to weight change = 1.2/0.5 Maximum weight change = 50
Bayesian Regularization (Br) (MCKAY, 1992)	Marquardt adjustment (*mu*) = 0.005 *mu* decrease/increase factor = 0.1/10 *mu* maximum value = 10^{10}
Levenberg-Marquardt (LM) (HAGAN and MENHAJ, 1994)	The same parameters and values as the method above.

Figure 5.9 presents a summary of results for the artificial neural network algorithms, for each centrality measure; ranked version with 99% confidence intervals. The algorithms were grouped by their category (first order, line search and second order), and then by increasing order of solution quality. Notice that Figure 5.9 shows only the results for the three-hidden layer

network architecture with 20 neurons in each layer – all the other architectures exhibited a similar pattern.

Figure 5.9 – Outline of Results for Different Training Algorithms



One can notice that in Figure 5.9, although the difference between most algorithms is considerably small, the second-order algorithms perform better than the ones with line search, which in their turn were generally better than the first-order algorithms. Moreover, my experiments have shown that despite the fact that the second order algorithms required a considerably larger time for each training epoch but they proved to be more efficient with a faster convergence, i.e., required a lower number of epochs to converge for our task.

The LM algorithm presents a slightly better overall result than all others do for Rank Closeness despite being statistically like (considering the confidence intervals) all second-order algorithms for Rank Betweenness. Therefore, the LM algorithm was selected to further improvements and experiments in the final stage of parameter optimizations.

In the third and final stage, I optimize the parameters within the LM method (Marquardt adjustment's initial value, decrease/increase factors and maximum value).

The Marquardt adjustment (mu) initial value is set to a value close to zero. This allows the training algorithm to apply larger weight updates and so speed up the initial convergence, due to the fact that the weight values are initially set randomly (with lower and maximum bounds to avoid an initial and detrimental saturation of the activation function). It is not very important what exact value is chosen considering that the algorithm will further adjust the size of the steps dynamically during the training with two other parameters: mu decrease and increase factors.

The LM method has also a parameter to set a maximum value for mu, which limits the method to a smallest step size and can avoid overfitting and the waste of time with finer grained, but nearly useless steps. In my application, this parameter can be set to an arbitrarily high value since I also use a maximum time limit to stop the training early and later (in the final setup and training) I use a validation set to avoid overfitting.

Thus, I focus on the optimization of the mu decrease/increase factors, which are responsible for adjusting the mu factor at each training iteration and are usually sensible to each specific application, requiring finer adjustments to achieve the best results in solution quality and training performance. The decrease factor parameter is a proportion of the increase factor, preferably smaller than the other to prevent loops during training. The exact number of the increase and decrease factors depends on the numerical amplitude of the training data. As they are not easy to estimate, I tested a wide range [1.5,100] of parameter combinations.

The results of this comparison for each centrality measure ranked version, with 99% confidence intervals, can be checked in Figure 5.10. Figure 5.10 also shows only the results for the three-hidden layer network architecture with 20 neurons in each layer – all the other architectures exhibited a similar pattern for this setup also.

We can note by looking at Figure 5.10 that excluding the decreasing factor value of 1 (100% of the increase factor), all other combinations of parameters were statistically similar within the confidence intervals. This fact demonstrates that these parameters do not interfere as much as one may think in this task. However, the combination 1.5 with 0.1 of increasing and decreasing factors, respectively, presents the lowest variance and the highest mean in my experimental results.

Hence, I trained my final model using a multilayer perceptron artificial three-hidden layer network with 20 neurons in each layer and the LM algorithm with initial/maximum mu set to 0.005/1010 and increasing/decreasing mu factor set to 1.5/0.1.

Figure 5.10 – Outline of Results for Each Combination of Parameters of the LM Algorithm

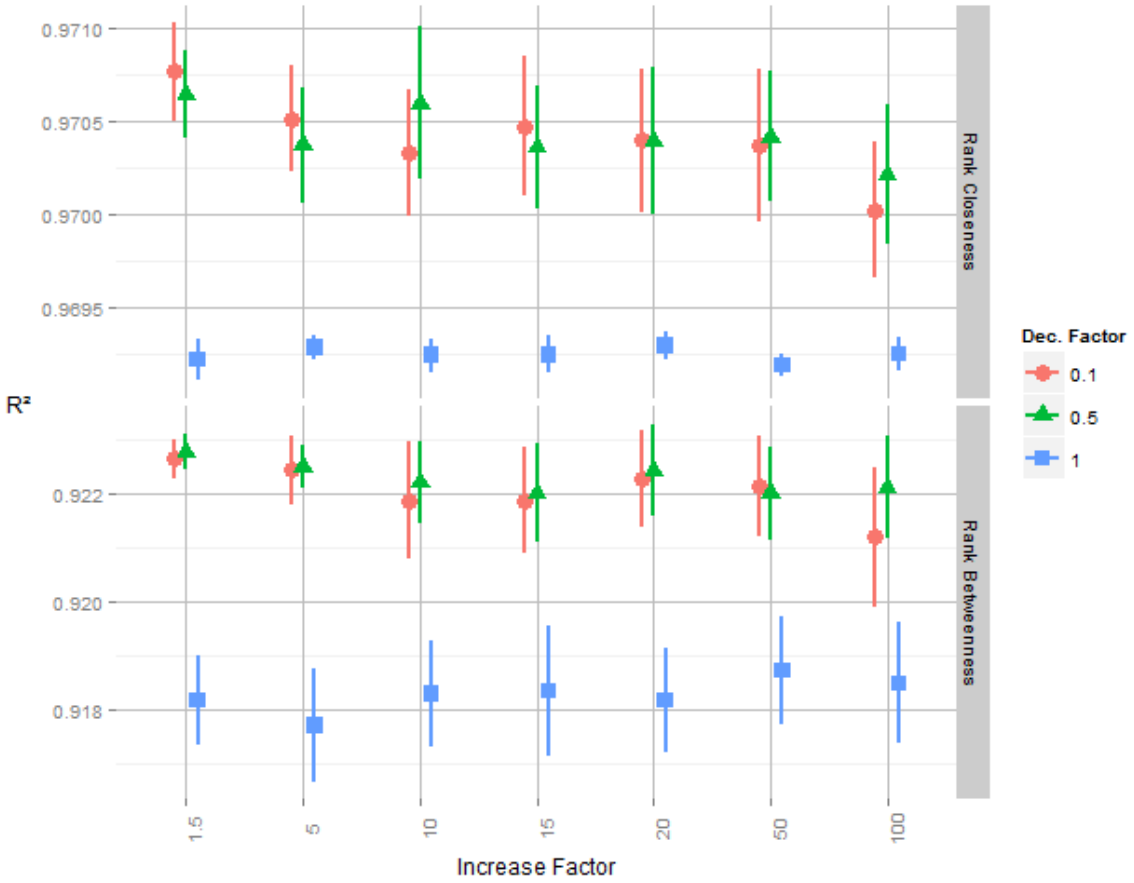
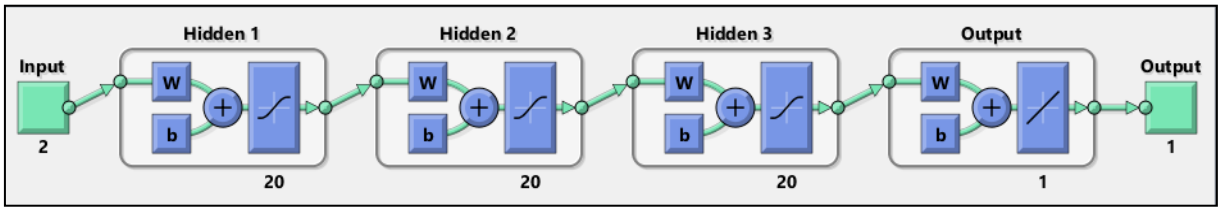


Figure 5.11 depicts and summarizes the artificial neural network structure visually as provided by the MATLAB tool.

Figure 5.11 – Artificial Neural Network Architecture using the LM algorithm



Finally, I tested a different input configuration with the addition of two-hop degree rank (a vertex degree value summed with the degree of its immediate neighbors). In this experiment I only tested the neural network configuration setup defined for the final model. The results showed that the addition of a new input improved the accuracy of the model to approximate closeness centrality by 5% on average, but it reduced the accuracy for betweenness centrality

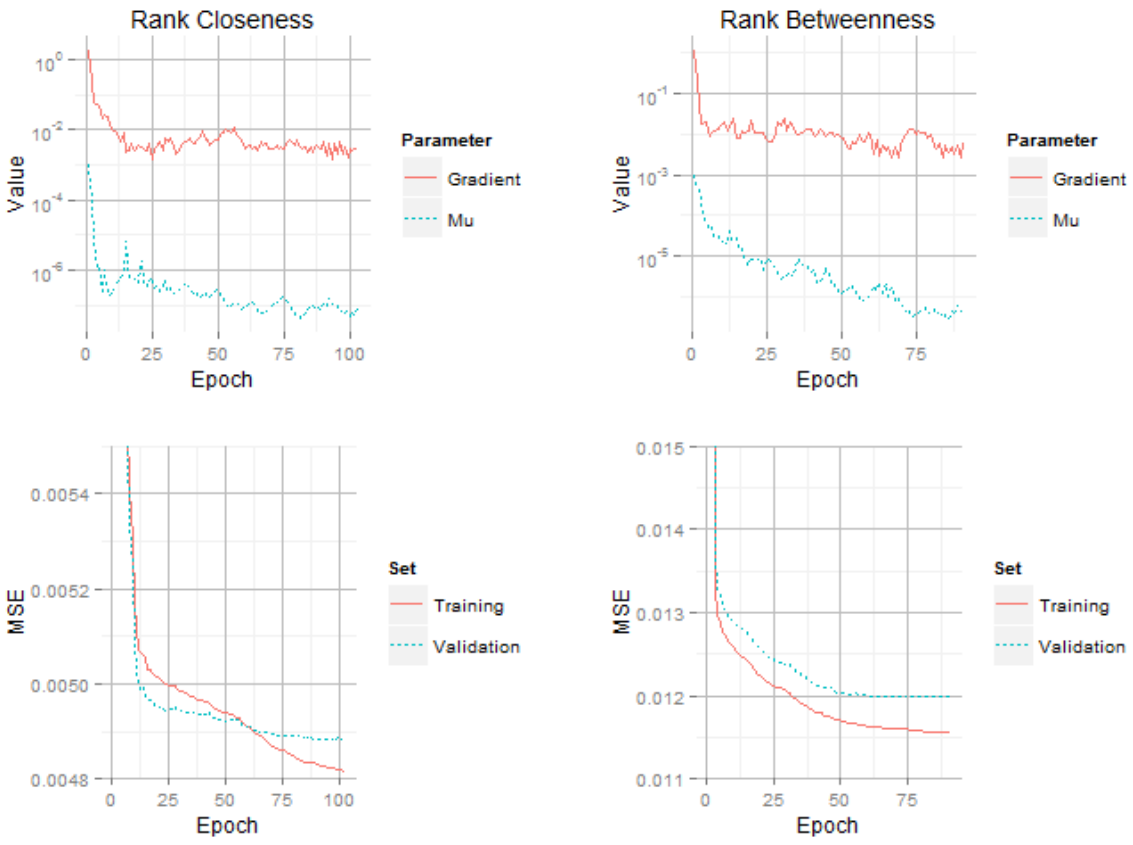
by 10% on average. Therefore, I chose to use just the two basic inputs (degree and eigenvector ranks) for the final training of the model.

For the setup that generates the final model, I divided the data uniformly at random (330,000 vertices from 600 different synthetic networks) into a set with 85% for training and a set with 15% for validation to prevent overfitting. The training stops whenever the solution does not improve for the validation set in 10 consecutive full batches with no training time limit. The best model result for the validation set is the one considered final (for such purpose the last 10 states of model are saved).

I run the experiments in the same machine specifications of the previous experiments with all CPU cores allocated for parallelism. The machine had the following specifications: Intel Core i7-5820K processor with six 3.3Ghz physical cores with 15MB shared cache memory, quad-channel 4x4GB DDR4 2133MHz RAM memory, Windows 10 operating system.

Figure 5.12 presents the training behavior of the parameters and the mean squared error (MSE) evolution over each epoch of training for each target centrality measure

Figure 5.12 – Parameters' Training Behavior



Notice that although MSE continues to drop, the validation set serves as an early stopping criterion to prevent overfitting, because it was not used for training. The total training time took about 10min for rank closeness and about 7min for rank betweenness.

The learning/regression of rank betweenness showed higher difficulty than rank closeness, even though both presented very low error bounds. This is supported by the final MSE values, one of each is half the other, but both are considerably small, since the output is in the interval [-1,1] and I used 330 thousand samples for training.

In the next subsection we compare the performance of the final setup for the artificial neural network with other common machine learning techniques in the literature and also available in the MATLAB tool environment.

5.3.3 Comparison Between Different Machine Learning Models

There are many machine learning techniques capable of creating regression models in tasks such as the ones tackled in this work. To reinforce the application of artificial neural learning models I compared their performance with other machine learning techniques from the literature and also available in the MATLAB environment. For such purpose I applied the same training data and machine configuration applied for the neural learning algorithm (depicted in subsections 5.3.1 and 5.3.2).

I applied a 10-fold cross-validation analysis to compute the R^2 as comparison quality factor and to optimize the other machine learning parameters. The results and the best configuration of each algorithms applied in my experiments are described in Table 5.5. Due to the robustness of the implementation of the algorithms in MATLAB, all the 99% confidence intervals lie in the fourth decimal place; therefore, they were not shown in the Table.

The column clarity in Table 5.5 stands for Hard for methods considered "*black box*" and Easy for the "*white box*" methods (methods that explicitly define how they interpret the inputs via a readable set of parameters). The column flexibility in Table 5.5 stand for the ability of each model to fit any kind of data. Therefore, high flexibility models can suffer more with overfitting while low flexibility models cannot be able to fit the data properly.

TABLE 5.5 – Machine Learning Models Comparison

Learning Algorithm	Description	Clarity	Flexibility	C_C R^2	C_B R^2
Linear Regression	A linear regression model with only intercept and linear terms	Easy	Very low	0.95	0.87
Interactions Linear	A linear regression model with intercept, linear and interaction terms	Easy	Medium	0.95	0.87
Robust Linear	A robust (less sensitive to outliers) linear regression model with only intercept and linear terms	Easy	Very low	0.95	0.86
Stepwise Linear	A linear model with terms determined by a stepwise algorithm	Easy	Medium	0.95	0.87
Fine Tree	A fine regression tree with minimum leaf size of 4	Easy	High	0.96	0.88
Medium Tree	A medium regression tree with minimum leaf size of 12	Easy	Medium	0.96	0.89
Coarse Tree	A coarse regression tree with minimum leaf size of 36	Easy	Low	0.95	0.89
Boosted Trees	An ensemble of regression trees using the LSBoost algorithm	Hard	Medium to High	0.95	0.87
Bagged Trees	A bootstrap-aggregated ensemble of regression trees	Hard	High	0.95	0.88
Linear SVM	A support vector that follows simple linear structure in the data, using the linear kernel	Easy	Low	0.95	0.87
Quadratic SVM	A support vector machine that uses the quadratic kernel	Hard	Medium	0.96	0.87
Fine Gaussian SVM	A support vector machine that follows finely-detailed structure in the data. It uses the Gaussian kernel with kernel scale $\sqrt{1/2}$	Hard	High	0.96	0.88
Medium Gaussian SVM	A support vector machine that finds less fine structure in the data. It uses the Gaussian kernel with kernel scale $\sqrt{2}$	Hard	Medium	0.96	0.87
Coarse Gaussian SVM	A support vector machine that follows coarse structure in the data. It uses the Gaussian kernel with kernel scale $4\sqrt{2}$	Hard	Low	0.96	0.87
MLP Neural Network with Backpropagation	A Multilayer Perceptron Neural Network Implementation with Backpropagation Learning	Hard	High	0.97	0.92

Notice that due to the high number of samples in the training data even small differences in the R^2 means a considerable disparity in the performance.

The results show that the neural network architecture performs considerably better than all other techniques to approximate Betweenness centrality and slightly better for Closeness centrality. It proved to be the more flexible (performed equally well for both centralities) while

robust methodology between the ones tested in my experiments. However, it may be hard to interpret how it works and how it computes the centrality measures based on the inputs.

The next step in my experiments was to compare the artificial neural network model's performance in real case scenarios with the computation of the exact centrality values and with the sampling approximation techniques. All of these will be further detailed and discussed in the next subsection.

5.3.4 Artificial Neural Network Learning with Real World Network Data

The final stage of a machine learning application is testing the respective learning algorithm/model with real world data to validate the model. In my experiments, I used 30 real-world networks from four freely available data repositories of network data. All the selected networks were symmetric and binary (unweighted edges). Only the largest connected component (LCC), which was computed for all analyzed networks, is used in the experimental validation.

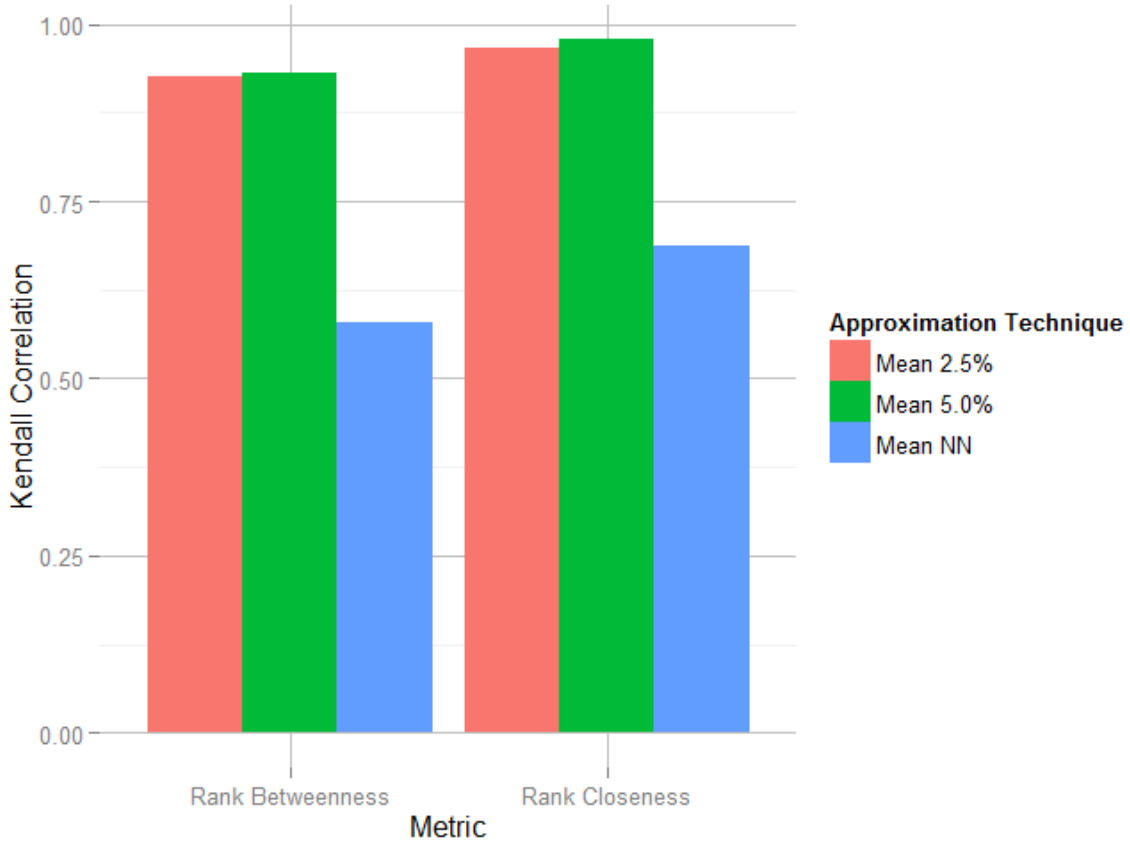
First, I computed eigenvector and degree centralities for all vertices of the real networks. The rank of the vertices in each network considering each of the centralities is then used as inputs for the machine learning model to approximate its rank in a target selected centrality (betweenness or closeness in my experiments). Notice that a different model is used depending on the target centrality measure because each model was trained specifically to approximate each one of the target centralities (subsection 5.3.2).

I also computed the exact values for the betweenness and closeness centralities for all the networks, but they are only used to compute the precision/error of the results provided by the approximation methods and to compare their cost in time (Table 5.2 of subsection 5.2).

The next stage is to run the model generated by the artificial neural network and previously trained to approximate betweenness and closeness for the vertices of the real networks. In order to do so, I used the same computer configuration of the training tasks in a parallel environment with 6 cores. The computation took less than 1s for any of the networks, which is a significant result and illustrates the effectiveness of the machine learning methodology. Then, I compared the results generated by the neural network (NN) with the results of the approximation algorithms (with sample sizes of 2.5% and 5.0%) using the Kendall τ-b correlation coefficient.

These results are summarized in Figure 5.13 and Figure 5.14. Figure 5.14 presents the results considering only three networks (Blog Catalog 3, Foursquare and Livemocha). These were the top 3 networks w.r.t. the model's performance for both centrality measures.

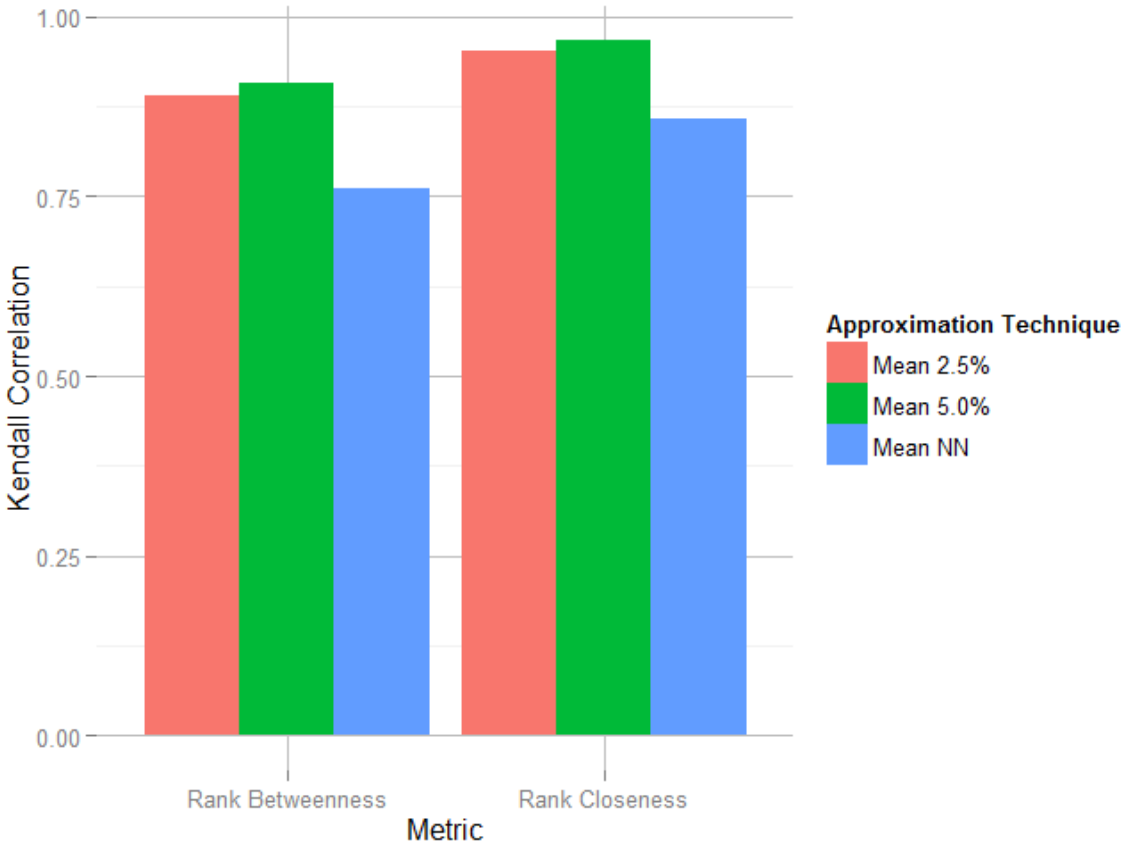
Figure 5.13 – Mean Correlation Coefficients Computed for All Real-World Networks



One can see that there is just a small gain for the approximation algorithms when the size of sample is doubled (notice the difference between 2.5% and 5.0%), which is hardly worth the cost considering that the computation time is doubled. The mean results obtained with the model for all networks were lower in quality comparing with the sampling methods, but still granted good results (over 0.55 correlation for betweenness and 0.65 for closeness centralities).

The results showed a great majority of variability: the model performed poorly for some of the networks (Euroroad, Power Grid, Facebook and Amazon for instance) with correlation coefficients lower than 0.4 and really well in others (Blog3 and Foursquare) with coefficients above 0.7.

Figure 5.14 – Mean Correlation Coefficients Considering Only the Top 3 Networks



I thought that this behavior is mainly caused by the data used during the training, which was generic for all networks and unable to fulfill specificities for some of the real networks. That is why the specific training data was generated and new experiments were carried on to confirm such hypothesis.

One can see that the learning model achieved a poorer result for the infrastructure networks (Euroroad, Texas, and US Power Grid for instance) and a better performance in social networks. Moreover, because of the generic formulations based mainly on social networks characteristics that were input for the BTER model (which generates the synthetic networks used for the artificial neural network training), I expected a large variance in the results presented by the model considering distinct kinds of networks. This variance in the results, however, is not observed for the sample-based algorithms, which perform nearly equally well in all networks tested. For such reason, I trained a specific neural network model for each of the six networks that presented the lower and higher results (Euroroad, Power Grid, Foursquare, Amazon, Blog3 and Facebook) with specific generated networks with the parameters provided by each of these networks.

I also tested multitasking neural networks capable of learning both centralities (closeness and betweenness) at the same time. Moreover, I tried to add another attribute for the training. In addition to the degree and eigenvector, I added a metric composed by the sum of the vertex degree with the degree of his neighbors (called second level or two-hop degree). The experiments comprised all the combinations of sizes of training set networks (2000 or 3000), three or just two attributes (addition second level degree or not) and the use of multitasking or not. I generated networks with the same size of the original network exclusively for the Euroroad real network due to its original size of less than 2000 vertices.

To simplify the visualization of the comparative analysis I used the code NN (baseline) for the neural network model trained with the generic dataset and the code NNTAM for the model trained with the specific training set. T assumes value 2 for the networks with size 2000 and 3 for the size 3000, A is the input attributes (1 – for degree and eigenvector; 2 – for two-hop degree and eigenvector; 3 – for all the three attributes) and M assumes 1 for simple tasking and 2 for multitasking networks.

First, I have compared and analyzed the correlation values between the approximation methods. Tables 5.6 and 5.7 summarizes the correlation results of some of the combinations tested with best neural network model for each network highlighted in gray. The other tests (3xx and 231) were omitted because they generated results similar to their counterparts.

The sampling techniques performed better than all neural models tested but the difference is minimal in some cases. I already expected that the sampling-based techniques performs better than the machine learning models simply because they have access to more information about the overall network structure with the drawback of requiring a lot more of computation time to acquire such information

TABLE 5.6 – Correlation Coefficients for Betweenness

Network	**Approximation Technique**							
	Sample		**Neural Network Model**					
	2.5%	**5.0%**	**NN**	**211**	**212**	**221**	**222**	**232**
Amazon	0.91	0.91	0.27	0.35	0.32	0.26	0.26	0.27
Blog3	0.89	0.90	0.73	0.80	0.80	0.71	0.70	0.71
Euroroad	0.86	0.88	0.16	0.41	0.41	0.44	0.44	0.46
Facebook	0.67	0.75	0.36	0.30	0.35	0.24	0.20	0.29
Foursquare	0.89	0.92	0.73	0.75	0.72	0.66	0.66	0.68
PowerGrid	0.93	0.92	0.21	0.57	0.51	0.43	0.42	0.45

TABLE 5.7 – Correlation Coefficients for Closeness

Network	Approximation Technique							
	Sample		Neural Network Model					
	2.5%	5.0%	NN	211	212	221	222	232
Amazon	0.99	0.99	0.10	0.42	0.52	0.66	0.65	0.65
Blog3	0.95	0.96	0.69	0.89	0.89	0.92	0.93	0.93
Euroroad	0.87	0.90	0.09	0.60	0.60	0.62	0.62	0.62
Facebook	0.87	0.93	0.26	0.38	0.34	0.35	0.36	0.43
Foursquare	0.93	0.94	0.48	0.85	0.84	0.88	0.88	0.88
PowerGrid	0.90	0.93	0.12	0.03	0.14	0.26	0.26	0.24

The neural models trained with generic networks performed considerably worse than the ones trained with the specific training set of networks excluding one case (Facebook network), where the generic model performed a little better. The difference among the neural models is greater on networks where the first model performed worse.

Amongst the parameters tested, I noticed that the size of the specific networks used for training the model was statistically irrelevant for the results considering 99% confidence intervals with 10 trials. This was also true in most cases for the multitasking.

The addition of a third attribute seems to contribute for the overall performance when approximating the closeness centrality, but sometimes it is harmful to approximate betweenness.

In the next step, I computed and analyzed the percentage of correctly classified vertices by their rank considering percentile sets of the network. For this analysis, I ordered the vertices by their exact centrality values, divided such set of vertices in percentiles (from 0.2% of the first ranked vertices to the first 25% vertices), and computed the percentage of these vertices for each percentile that appears in the same percentile in each approximation technique. A 100% match means that every element from one set is on the other (perfect classification), while 0% means that both sets are completely disjoint. Such kind of analysis is important to give us an idea of how close/distant the rankings of the vertices are considering only the more important vertices of the network (generally speaking, the ones of interest for most applications). It also shows us better and gives insights “where” the mistakes are, fact that is obscured when considering only the correlation coefficients.

Figure 5.15 presents the effect of different sampling sizes considering the mean results over the six networks analyzed. It only reinforces what the correlation values already show

(Tables 5.6 and 5.7). The upgrade in solution quality for the 5% sample is minimal comparing that it costs twice in terms of computational time.

Figure 5.15 – Percentiles Accuracy Comparison Between Different Sampling Sizes

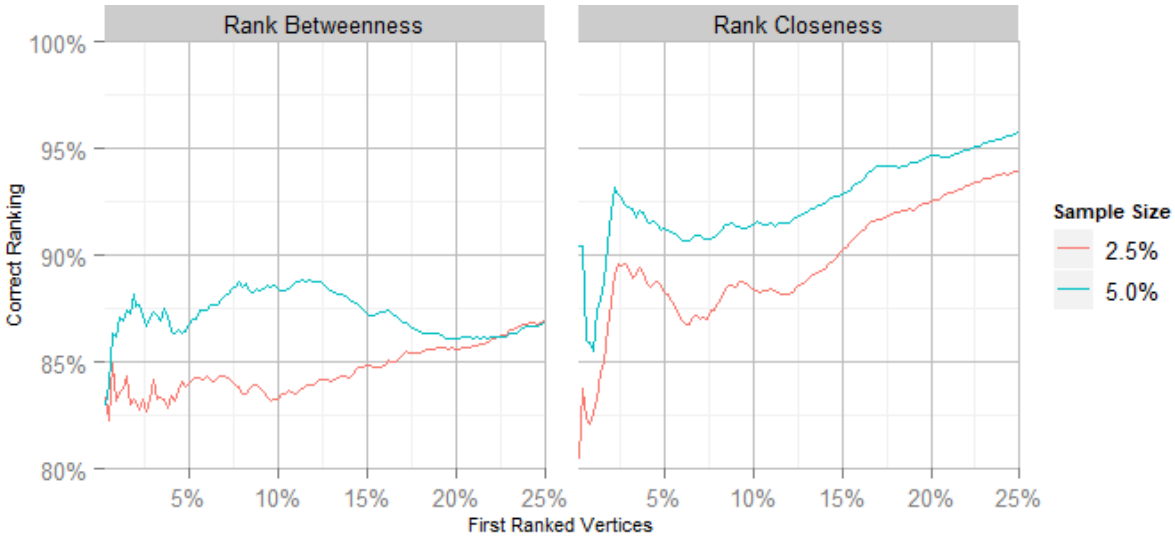


Figure 5.16 through 5.21 compares the results of the approximation methodologies for each network. Only the best approximation techniques (5% sample and best neural models) are showed to facilitate the reading of the figures.

Figure 5.16 – Correctly Classified Vertices by Percentiles for Amazon Network

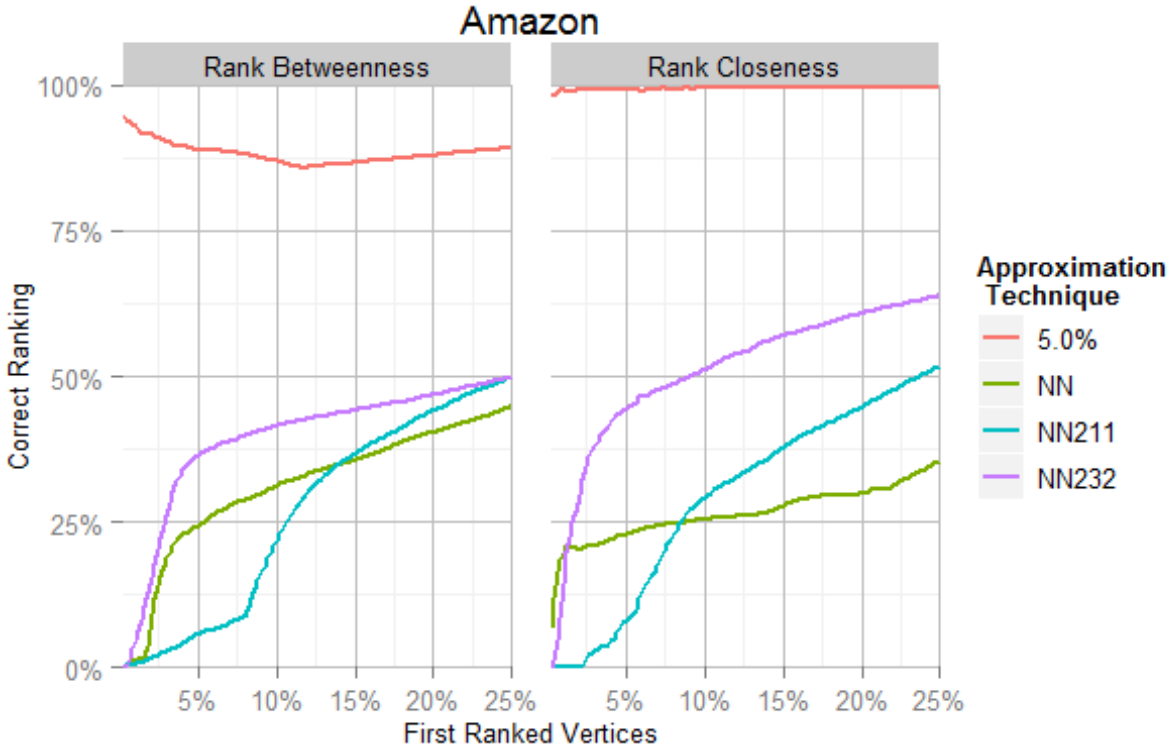


Figure 5.17 – Correctly Classified Vertices by Percentiles for Blog3 Network

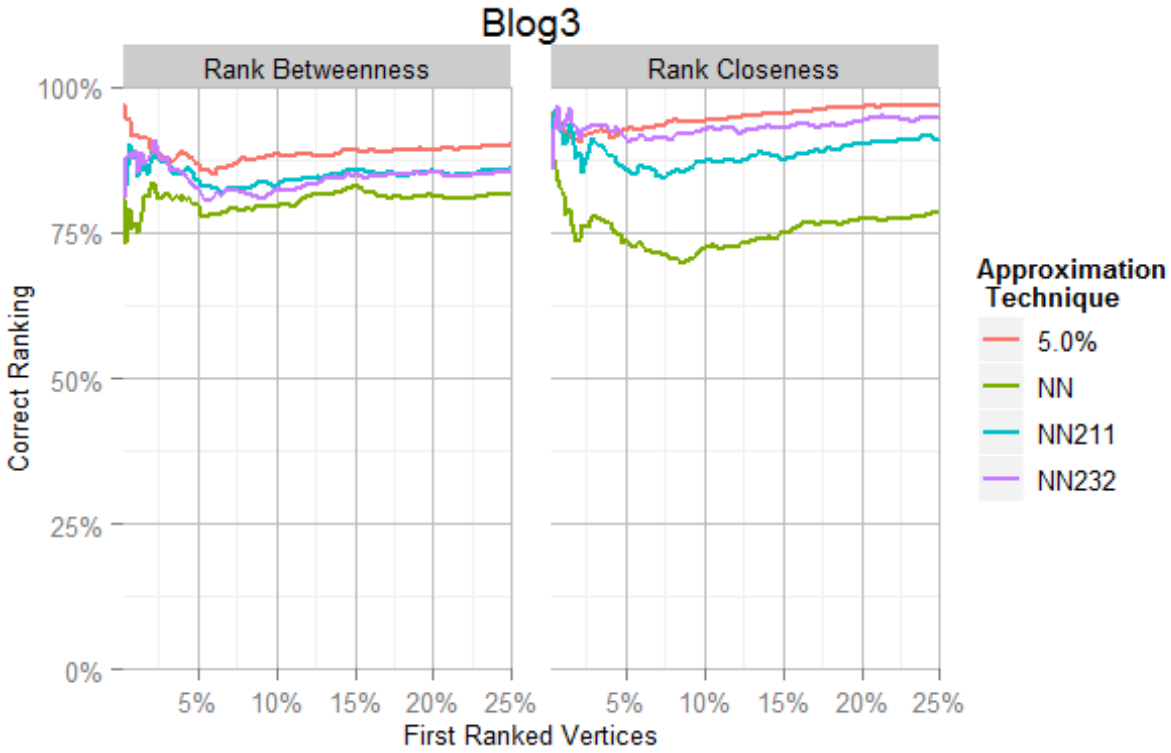


Figure 5.18 – Correctly Classified Vertices by Percentiles for Euroroad Network

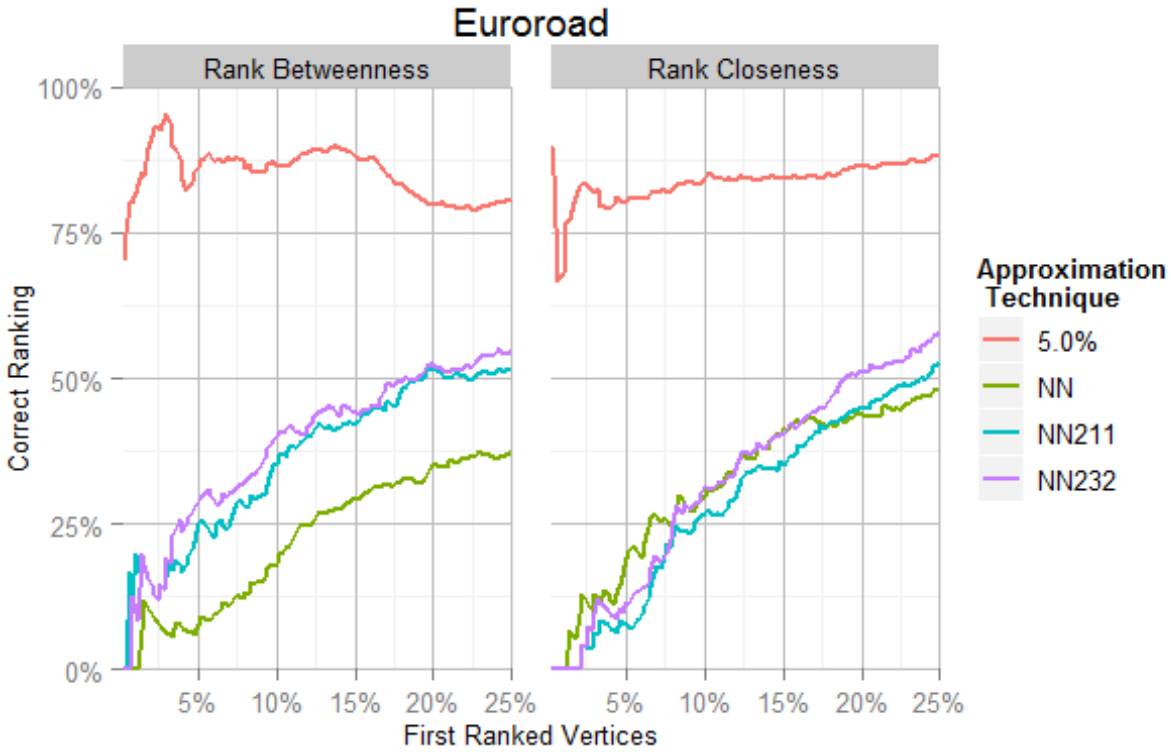


Figure 5.19 – Correctly Classified Vertices by Percentiles for Facebook Network

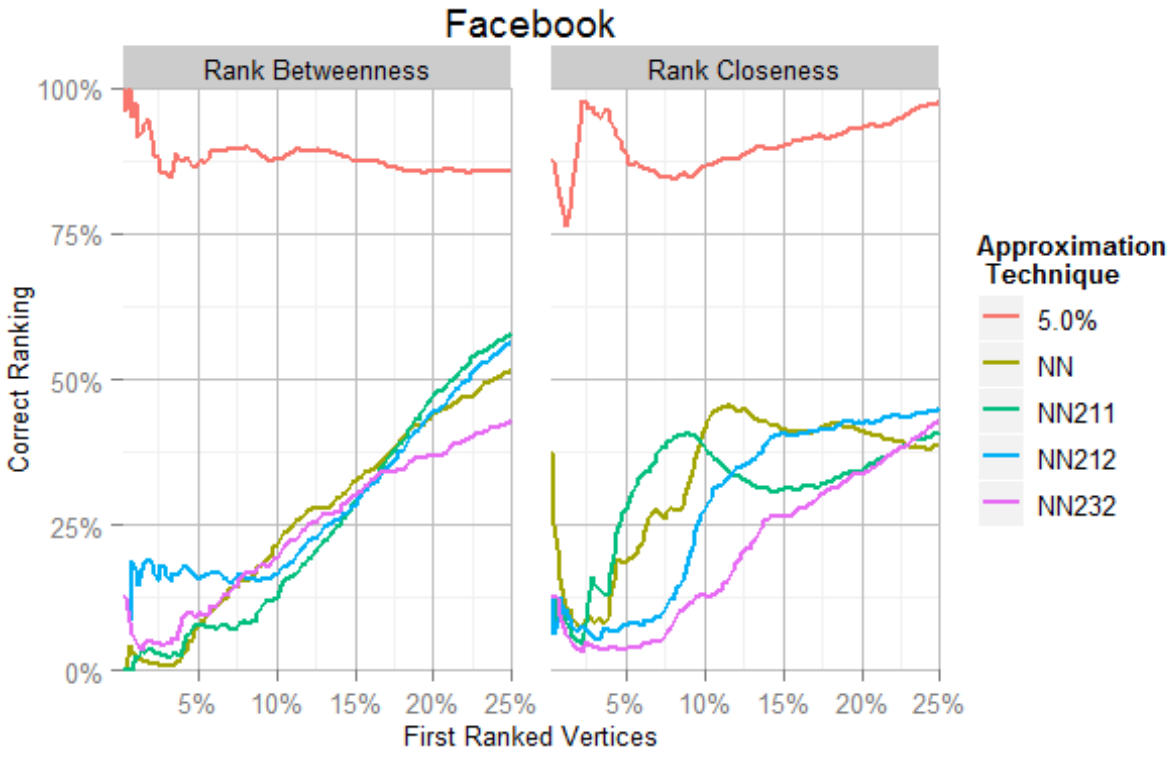


Figure 5.20 – Correctly Classified Vertices by Percentiles for Foursquare Network

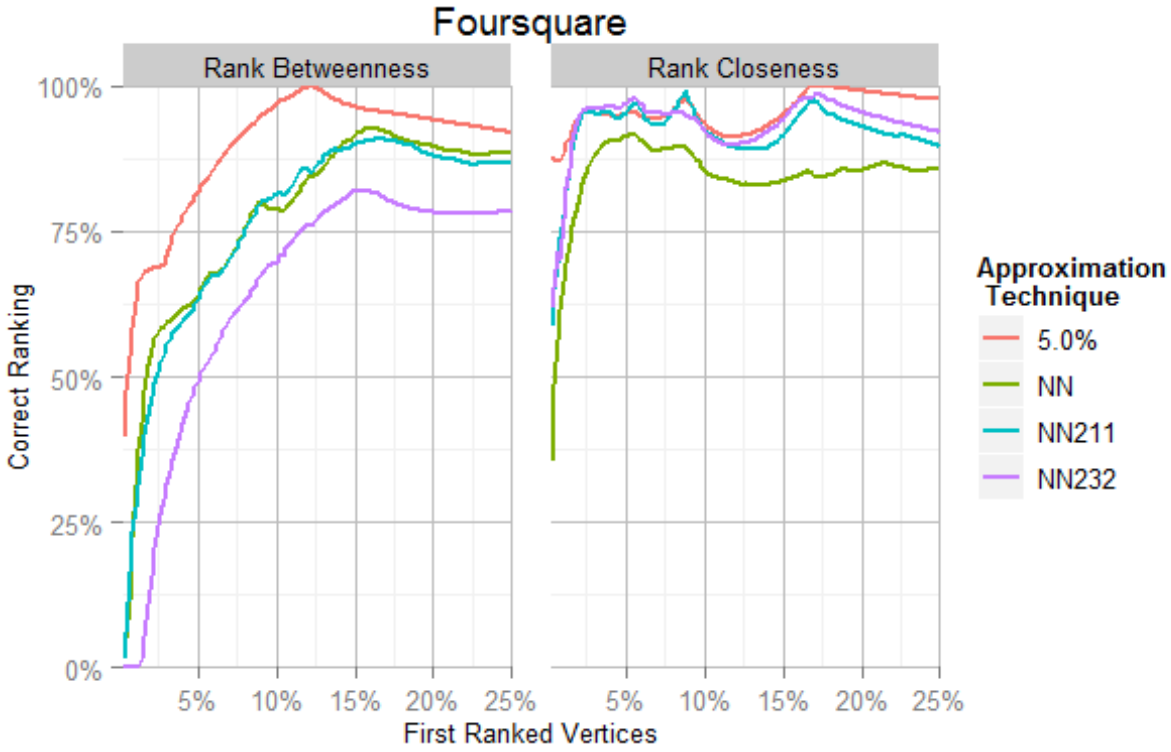
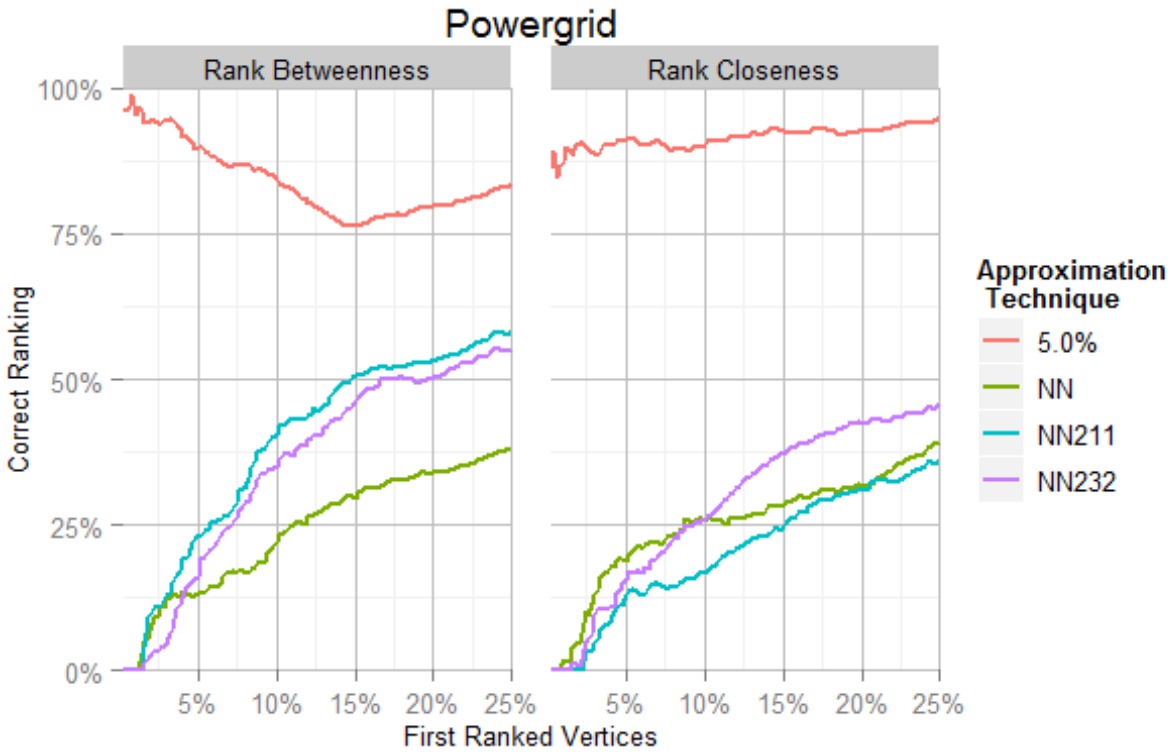


Figure 5.21 – Correctly Classified Vertices by Percentiles for Powergrid Network



We can see that the Blog3 and Foursquare networks were the only case scenario where the neural models performed as good as or a little better that the sampling methodologies for closeness centrality and near as good for betweenness. These networks probably have a simpler structure, therefore the information contained in the degree and eigenvector centralities (used as training inputs) fully characterize the network structure while in the other networks more complex information is needed to achieve a better performance.

In addition, the NN232 (i.e., multitasking network with three input attributes) configuration seems better than the other NN architectures to rank the first 25% ranked vertices except for betweenness centrality in Foursquare network and for both centralities in Facebook network.

One should also consider that the neural learning model is capable to compute the centrality for all vertices of a given network in seconds (even for massive networks) and that the sampling techniques takes at least some minutes for the smaller networks and hours or even days for the biggest networks.

Lastly, I analyzed visually the results obtained by the approximation neural model by drawing the representative graph of the Euroroad network.

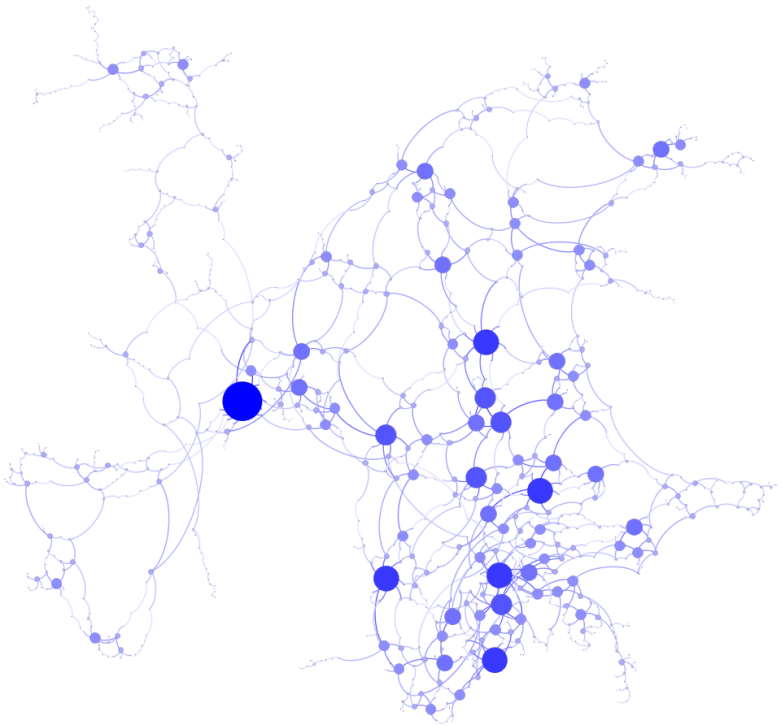
5.3.5 Visual Graph Analysis of the Capabilities of the Neural Model

I generated comparison graphs (using the software Gephi and the algorithm Force Atlas 2) for the Euroroad network to see the behavior of each centrality measure and the artificial neural network model.

I was unable to do the same visual analysis for the other real networks due to their size. It was impossible to organize properly and visually clear all the vertices so one can interpret the graphs.

Figure 5.22 shows the Euroroad network vertices and edges with an emphasizes on its community structures (which is due to the use of Force Atlas 2 algorithm to organize the vertices in the graph). The sizes of the vertices and the intensity of the color are correlated (logarithmic scaled for clarity) to the vertices' degree.

Figure 5.22 – Euroroad Network with Vertices Sized and Colored by their Degree



The first figure serves only as basis to interpret the comparisons that will be realized in the following figures. All the subsequent figures will retain the same vertices' position and

color from the Figure 5.22 but, the size of the vertices will be related to another centrality measure.

Figure 5.23 sizes the vertices in the graph by the two-hop degree, Figure 5.24 by the eigenvector centrality, Figure 5.25 by the betweenness centrality and Figure 5.26 by the closeness centrality.

All the centrality values used in Figures 5.23 to 5.26 were the exact values computed with the exact algorithm of each centrality. These figures will serve as basis to compare the input information, provided by the degree, two-hop degree and eigenvector centralities (respectively, Figure 5.22 above and subsequent Figures 5.23 and 5.24), with the desired output, i.e. target values, of betweenness and closeness centralities (correspondingly to the Figures 5.25 and 5.26).

Figure 5.23 – Vertices Sized by their Two-Hop Degree

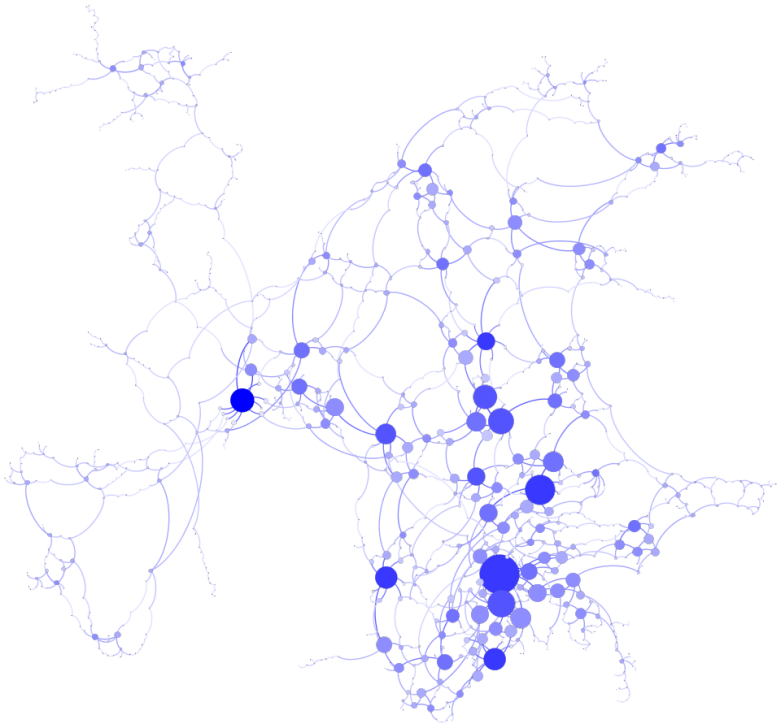


Figure 5.24 – Vertices Sized by their Eigenvector Value

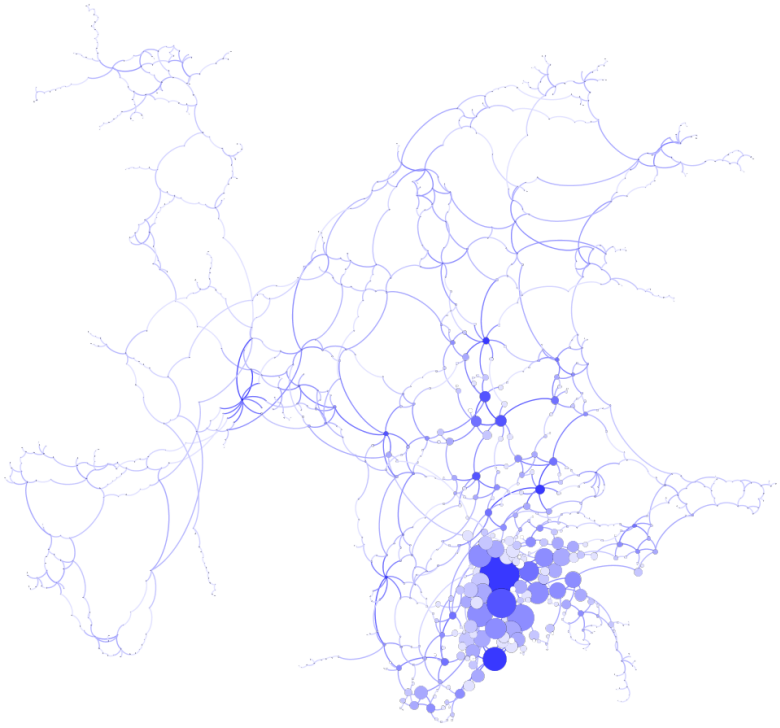
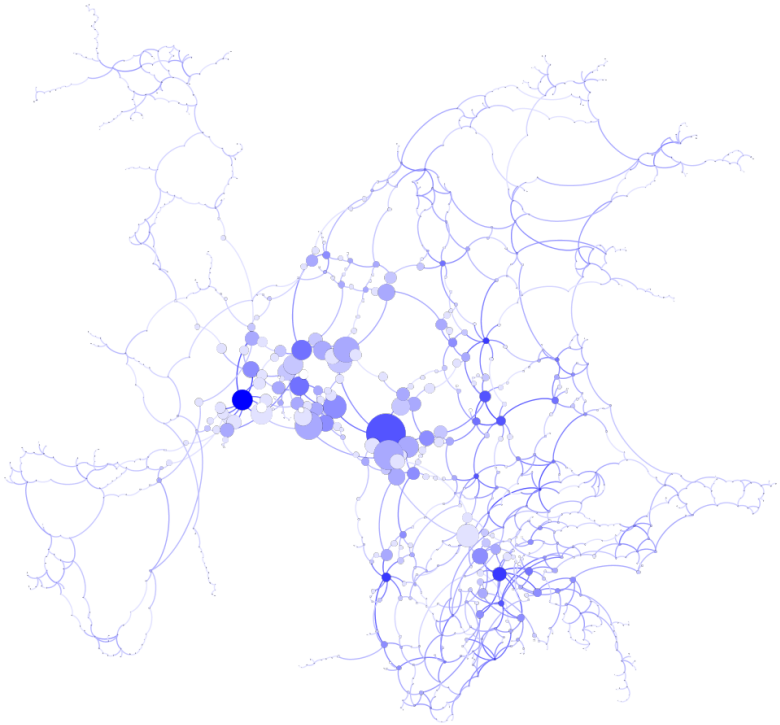


Figure 5.25 – Vertices Sized by their Betweenness Value

Figure 5.26 – Vertices Sized by their Closeness Value



As one can see in Figures 5.22, 5.23 and 5.24, the two-hop degree is a mid-term between degree and eigenvector centralities. We can also notice the main differences amongst the betweenness, closeness and input centralities. Betweenness leverages more paths and control (the more exclusive the path the better for the vertex centrality value because it has control by many others) while closeness considers more important the independency of a vertex by taking in consideration only the distances it has to all others. It is also noticeable that closeness is easier to predict with the input centralities than it is betweenness because the higher similarity that closeness has with the input centralities.

Next, I present the Figures 5.27 and 5.28 that show the results of the model trained with the generic synthetic networks (NN at Tables 5.6 and 5.7, and at Figure 5.18) for the approximation of betweenness and closeness centrality respectively.

Figure 5.27 – Approximated Betweenness

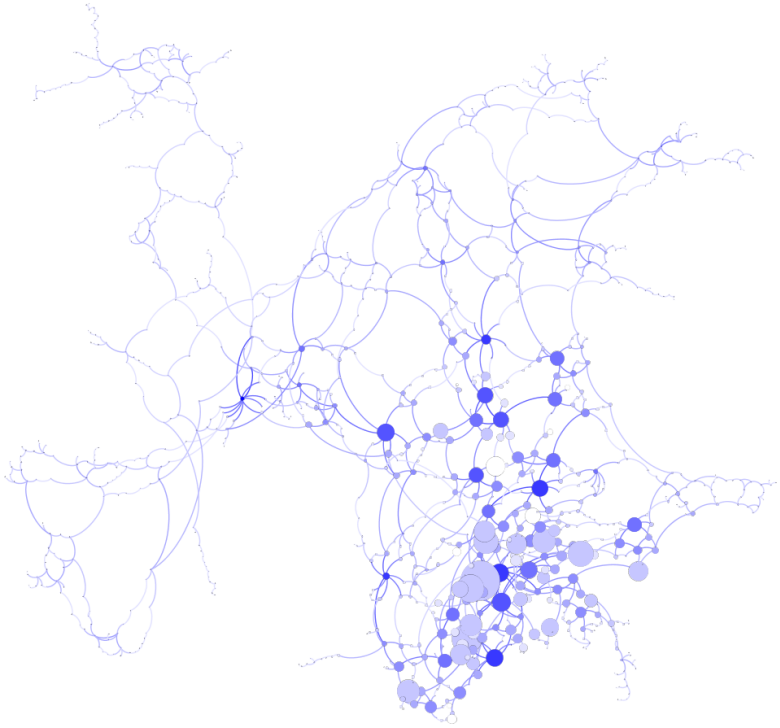
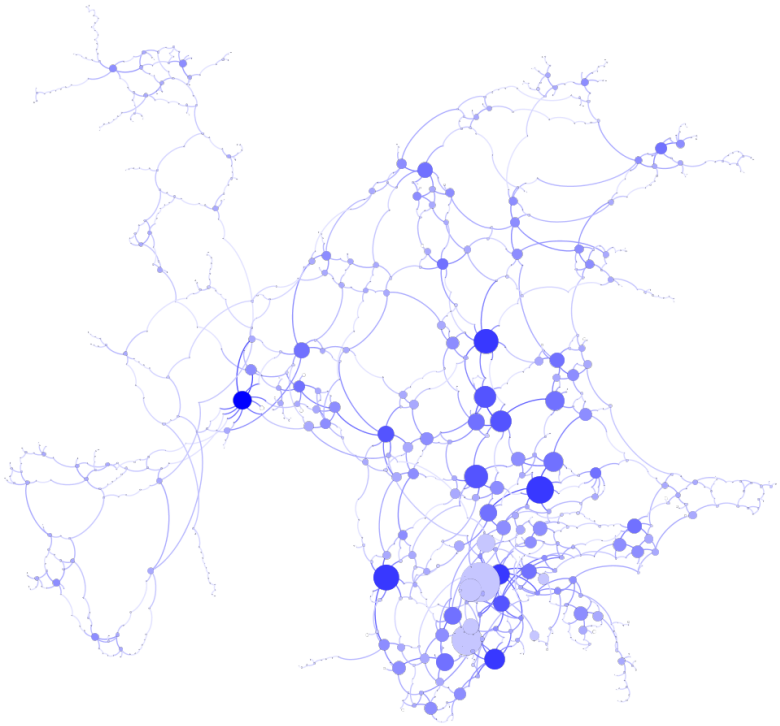


Figure 5.28 – Approximated Closeness



We can see that the results for some vertices both in closeness and betweenness are distant from the ideal but the approximation method still captures some aspects of the target centralities even though it has few information provided by the input centralities considering the expected target results.

For instance, several highly central vertices for betweenness centrality, the belt of vertices in the northwest part of the graph in Figure 5.25, are considered peripherical by all the three input centralities. Similarly, it happens for closeness centrality at the narrow part of the graph at the center of the figures. To improve such results, remember that I have generated and tested specific synthetic networks to train the neural model. Figures 5.29 and 5.30 show the results of the NN232 model which numeric analysis is summarized in Tables 5.6 and 5.7, and Figure 5.18.

Figure 5.29 – Approximated Betweenness with the NN232 Model

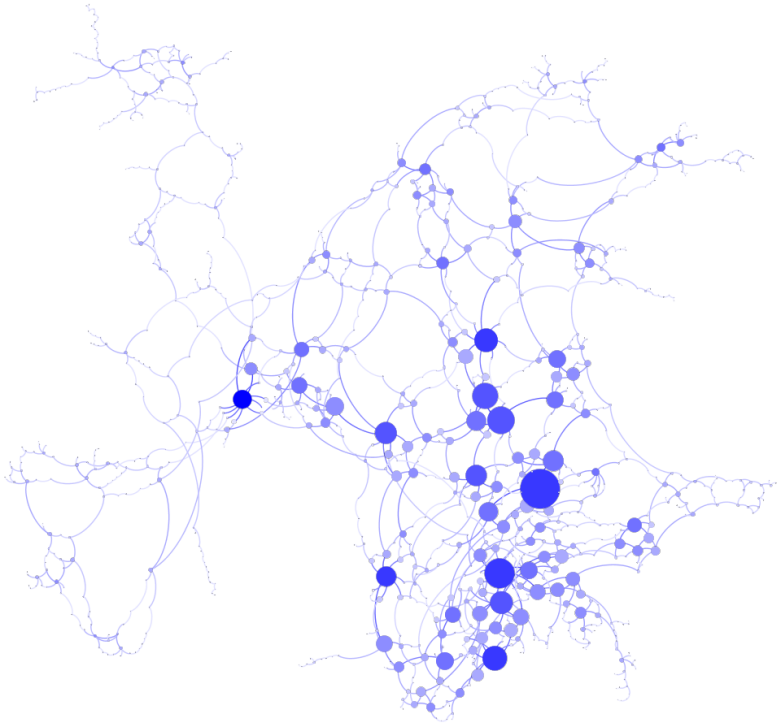
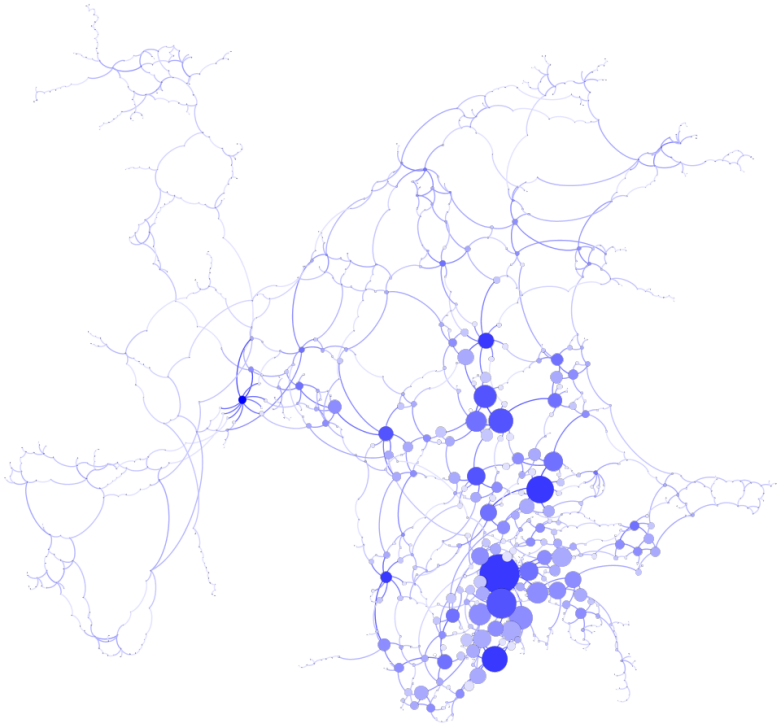


Figure 5.30 – Approximated Closeness with the NN232 Model



A considerable improvement is perceptible in many parts of the network for both target centralities where the previous model was incapable to capture such nuances in the graph substructures with the generic training data. It is also imperative to remember that the model performance in the Euroroad network was one of the lowest amongst all the 30 real scenarios tested. Fact that can be checked in Tables 5.6 and 5.7 and also confirmed by comparing the Figures 5.16 to 5.21. So, a good disparity between the results obtained (Figures 5.29 and 5.30) and the expected target values (Figures 5.25 and 5.26) were already expected. Nevertheless, the model still was unable to correctly assign the betweenness centrality values for the northwest belt of vertices and assign the correct closeness in the central area of the figures.

More importantly, with such analysis we can perceive that the input information (Figures 5.22 to 5.24) lacks enough evidence to the model in such kind of network structures. Consequently, additional information needs to be provided for the model to improve its performance even further. Despite all that, it is important to remember that the model was trained with networks far smaller than the real ones and that results shown a quite robust and generalizable model for most of the real cases scenarios.

All these results show that even though the neural network model was not capable to rank all the vertices in the correct order, it is effective enough to classify chunks of vertices as highly central and has a close performance with respect to the sample approximation methods that require considerably more computational time.

Many applications of centrality measures are interested in the selection of the top central vertices and not in their exact centrality order. For instance, the selection of influencers to optimize advertisement, the control and protection of critical spots in a communication network and the allocation of resources to enhance their distribution usually make use of a group of elements and do not rely on their specific order.

Moreover, these kind of networks and applications, massive and constantly evolving, require continuous analysis and the computation of such metrics become quite expensive in terms of computing resources. Therefore, the time and machine processing power saved by the use of a machine learning model is quite relevant in such applications and environments.

Still, the use of machine learning may not be adequate to approximate centralities in applications that rely on the exact centrality values or are oversensitive to the order of the centrality rank.

Ultimately, the machine learning techniques, especially the ones demonstrated and developed in this work for the approximation of centrality measures, are more adequate for applications where the use of centrality measures offers complementary analysis or are used as part of a decision process or heuristics.

6 CONCLUSION

The growing relevance of network research and applications demand the development of appropriate tools and methods for network analysis. These methods include vertex centrality measures which are widely used in many application domains. However, as networks grow in size, availability and relevance in several fields, their analysis and computational costs present challenges which may hinder some important applications and studies.

Machine learning techniques have recently been successful in a number of relevant applications tackling large amounts of data (GARCEZ et al., 2015; HAGAN and MENHAJ, 1994; LECUN et al., 2015). Moreover, the referred growing availability of massive network databases demands the use of effective techniques to better exploit and to interpret and take advantage of these data in an efficient, robust and scalable manner.

This work aimed at providing an effective approximation technique for the computation of centrality measures and, consequently, enabling their proper use in important applications with massive and/or dynamic networks by saving computational resources and providing an alternative solution in a short time. But my main objective it not restricted to that area, the main idea behind it is to aid developers and scientists by creating, validating and testing new strategies to tackle the complexity inherent to the efficient computation of graph analysis metrics in massive and dynamic networks.

I started by the principle that the more complex metrics are highly correlated with the simpler ones. Fact that was also empirically analyzed with the set of the eight centrality measures that are most commonly applied in the literature.

My experimental results revealed that two simple centralities, eigenvector and degree (and also the derivate two-hop degree version), can be used as input vector to approximate closeness and betweenness centralities although it proved to provide insufficient information in some cases for a precise and more accurate approximation. Likewise, using the two-hop degree as input information for the model seems to prejudicial or with little significance in some of the real scenarios tested.

In this context, I presented a comprehensive empirical analysis on the use of artificial neural networks learning to estimate centrality measures. I have tested and identified the best configuration for the artificial neural network training, including network structure, training algorithm, training meta-parameters and training data.

By means of extensive experimentations I found out that the multitasking network structure with three hidden-layers, 20 neurons in each layer, trained with the LM algorithm

(initial/maximum mu set to 0.005/1010 and increasing/decreasing mu factor set to 1.5/0.1) acquire the most promising results in the approximation of centrality measures (Figures 5.8, 5.9, 5.10 and Tables 5.6 and 5.7).

Nonetheless, the experimental results show us that the machine learning methodology is sufficiently general and robust for both centralities tested and it can be easily adapted and used with any other similar metric as targeted value, which increases their potential use.

It is also exposed that the neural network model is able to approximate the target centrality measures with considerable accuracy and reduced computational costs in 30 real-world experimental case scenarios (Figure 5.13 and 5.14). I have shown also how one can use the BTER generative complex network model as a way to provide unlimited and multipurpose training data as it may be configured to generate synthetic networks of any size and, at the same time, by preserving similar structural patterns of real networks present in any kind of application or environment.

The potential of the BTER model and its characteristics were tested by a series of experiments with a varied set of parameters (being represented by a different degree distribution functions and clustering coefficients). I have presented the primal advantage of the BTER model, its formidable capability to match a wide-ranging of degree distributions, and also its main flaw, the difficulty to parametrized correctly the clustering coefficient of the network being generated.

The methodology, algorithms and approximation models were tested in real-world case scenarios where I also generated networks with specific parameters to enhance the results of the previous models. The performance of the neural model improved considerably with this approach (Figures 5.16 to 5.21). The research also shows that the data used for training the model is one major factor that affects the learning model. Therefore, is fundamental to invest resources and time to provide a representative set of training data. Considering this result, one should always use the knowledge (degree distribution and clustering coefficients) about the network of interest to generate specialized training data; this will lead to improvements in the performance of the artificial neural learning model.

The methodology based on the artificial neural network model showed a noticeable and clear advantage and tradeoff with respect to computational costs, making it a viable option for applications where accuracy is not the solo goal, but in scenarios and configurations in which computation resources are limited. In such common situations, approximations via machine learning are an effective alternative, in particular in the context of large-scale complex network applications.

In summary, my work accomplished several tasks and objectives by a series of experiments:

(i) I associated and analyzed a set of eight centrality measures which resulted in the selection of the degree and eigenvector centralities as inputs attributes to approximate the more complex ones and by the preliminary selection of betweenness and closeness centralities as the more important targets for the experiments;

(ii) I compared and investigated the most widespread complex network models in the literature which resulted in the selection of the BTER model to generate synthetic networks with diminished size but still preserving the structural properties of real massive networks fact that is availed to generate plentiful and meaningful training data for the machine learning algorithms;

(iii) I tested and compared several approximation methodologies for the centrality measures, starting with the sampling techniques and then with several machine learning algorithms, including decision trees, SVMs and artificial neural networks. The results led to the choice and further testing and improvements for the artificial neural network structure;

(iv) I optimize the artificial neural network models (MLP) by empirically determining the best meta-parameters values for such task besides verifying its use in a varied set of synthetic and real cases scenarios comprising networks from several fields and with several sizes ranging to just a thousand vertices until about a million vertices;

(v) The main results of this work were published in important conferences and journals in the area of computer science and the field of artificial intelligence. This work resulted in four A1 (CAPES foundation evaluation) conference papers (two for IJCNN, one at ISCC and one at GLOBECOM) and in one A1 journal paper (ACMCSUR).

I believe that further improvements and work would be possible with the test and use of adversarial networks, graph neural networks and the study and selection of additional informative attributes as input for the model. These will consequently improve the accuracy of the artificial neural learning model and possibly open new opportunities for applications in other areas were accuracy plays a more critical role.

Suggested future research avenues include the approximation of temporal centrality measures using recurrent neural models and the development of tools capable of automatically

generating/training artificial neural networks able to approximate several centrality measures which would also facilitate further applications for this method in many areas and for many studies. The methodology presented in this work elicits the requirements towards building such tools.

REFERENCES

ATZORI, Luigi; IERA, Antonio; MORABITO, Giacomo. From: The next evolutionary step of the internet of things**. Ieee Communications Magazine**, [s.l.], v. 52, n. 1, p.97-105, jan. 2014. Institute of Electrical and Electronics Engineers (IEEE). http://dx.doi.org/10.1109/mcom.2014.6710070.

BADER, David A. et al. Approximating Betweenness Centrality. **Algorithms And Models For The Web-graph**, [s.l.], p.124-137, 2007. Springer Berlin Heidelberg. http://dx.doi.org/10.1007/978-3-540-77004-6_10.

BAIG, Mirza Basim; AKOGLU, Leman. Correlation of Node Importance Measures. **Proceedings Of The 24th International Conference On World Wide Web - Www '15 Companion**, [s.l.], p.275-281, 2015. ACM Press. http://dx.doi.org/10.1145/2740908.2743055.

BALDESI, Luca; MACCARI, Leonardo; LOCIGNO, Renato. Improving P2P streaming in Wireless Community Networks. **Computer Networks**, [s.l.], v. 93, p.389-403, dez. 2015. Elsevier BV. http://dx.doi.org/10.1016/j.comnet.2015.09.024.

BARABÁSI, A.. Emergence of Scaling in Random Networks. **Science**, [s.l.], v. 286, n. 5439, p.509-512, 15 out. 1999. American Association for the Advancement of Science (AAAS). http://dx.doi.org/10.1126/science.286.5439.509.

BASTUG, Ejder; BENNIS, Mehdi; DEBBAH, Mérouane. Living on the edge: The role of proactive caching in 5G wireless networks. **Ieee Communications Magazine**, [s.l.], v. 52, n. 8, p.82-89, ago. 2014. Institute of Electrical and Electronics Engineers (IEEE). http://dx.doi.org/10.1109/mcom.2014.6871674.

BASTUG, Ejder; HAMIDOUCHE, Kenza; SAAD, Walid; DEBBAH, Mérouane. Centrality-Based Caching for Mobiel Wireless Networks. **1st KuVS Workshop on Anticipatory Networks**, p.1-3, Stuttgart, Germany, 2014.

BATTITI, Roberto. First- and Second-Order Methods for Learning: Between Steepest Descent and Newton's Method. **Neural Computation**, [s.l.], v. 4, n. 2, p.141-166, mar. 1992. MIT Press - Journals. http://dx.doi.org/10.1162/neco.1992.4.2.141.

BAVELAS, Alex. A Mathematical Model for Group Structures. **Human Organization**, [s.l.], v. 7, n. 3, p.16-30, jul. 1948. Society for Applied Anthropology. http://dx.doi.org/10.17730/humo.7.3.f4033344851gl053.

BOCCALETTI, S et al. Complex networks: Structure and dynamics. **Physics Reports**, [s.l.], v. 424, n. 4-5, p.175-308, fev. 2006. Elsevier BV. http://dx.doi.org/10.1016/j.physrep.2005.10.009.

BOGUÑÁ, Marián et al. Models of social networks based on social distance attachment. **Physical Review e**, [s.l.], v. 70, n. 5, p.1-10, 22 nov. 2004. American Physical Society (APS). http://dx.doi.org/10.1103/physreve.70.056122.

BONACICH, Phillip. Factoring and weighting approaches to status scores and clique identification. **The Journal Of Mathematical Sociology**, [s.l.], v. 2, n. 1, p.113-120, jan. 1972. Informa UK Limited. http://dx.doi.org/10.1080/0022250x.1972.9989806.

BONATO, Anthony. A Survey of Properties and Models of Online Social Networks. **International Conference on Mathematical and Computational Models**, p.1-10, New Delhi, India, 2009.

BORGATTI, Stephen P.; CARLEY, Kathleen M.; KRACKHARDT, David. On the robustness of centrality measures under conditions of imperfect data. **Social Networks**, [s.l.], v. 28, n. 2, p.124-136, maio 2006. Elsevier BV. http://dx.doi.org/10.1016/j.socnet.2005.05.001.

BRANDES, Ulrik. A faster algorithm for betweenness centrality*. **The Journal Of Mathematical Sociology**, [s.l.], v. 25, n. 2, p.163-177, jun. 2001. Informa UK Limited. http://dx.doi.org/10.1080/0022250x.2001.9990249.

BRANDES, Ulrik. On variants of shortest-path betweenness centrality and their generic computation. **Social Networks**, [s.l.], v. 30, n. 2, p.136-145, maio 2008. Elsevier BV. http://dx.doi.org/10.1016/j.socnet.2007.11.001.

BRANDES, Ulrik; PICH, Christian. CENTRALITY ESTIMATION IN LARGE NETWORKS. **International Journal Of Bifurcation And Chaos**, [s.l.], v. 17, n. 07, p.2303-2318, jul. 2007. World Scientific Pub Co Pte Lt. http://dx.doi.org/10.1142/s0218127407018403.

BUTTS, Carter T.. Exact bounds for degree centralization. **Social Networks**, [s.l.], v. 28, n. 4, p.283-296, out. 2006. Elsevier BV. http://dx.doi.org/10.1016/j.socnet.2005.07.003.

CAO, Xianbin et al. Pedestrian detection in unseen scenes by dynamically updating visual words. **Neurocomputing**, [s.l.], v. 119, p.232-242, nov. 2013. Elsevier BV. http://dx.doi.org/10.1016/j.neucom.2013.03.036.

CHARALAMBOUS, C.. Conjugate gradient algorithm for efficient training of artificial neural networks. **Iee Proceedings G Circuits, Devices And Systems**, [s.l.], v. 139, n. 3, p.301-310, 1992. Institution of Engineering and Technology (IET). http://dx.doi.org/10.1049/ip-g-2.1992.0050.

CHEN, Yiling et al. Mathematical foundations for social computing. **Communications Of The Acm**, [s.l.], v. 59, n. 12, p.102-108, 1 dez. 2016. Association for Computing Machinery (ACM). http://dx.doi.org/10.1145/2960403.

CHO, Eunjoon; MYERS, Seth A.; LESKOVEC, Jure. Friendship and mobility. **Proceedings Of The 17th Acm Sigkdd International Conference On Knowledge Discovery And Data Mining - Kdd '11**, [s.l.], p.1082-1090, 2011. ACM Press. http://dx.doi.org/10.1145/2020408.2020579.

CHUNG, F.; LU, L.. The average distances in random graphs with given expected degrees. **Proceedings Of The National Academy Of Sciences**, [s.l.], v. 99, n. 25, p.15879-15882, 4 dez. 2002. Proceedings of the National Academy of Sciences. http://dx.doi.org/10.1073/pnas.252631999.

CHUNG, Fan; LU, Linyuan. Connected Components in Random Graphs with Given Expected Degree Sequences. **Annals Of Combinatorics**, [s.l.], v. 6, n. 2, p.125-145, nov. 2002. Springer Nature. http://dx.doi.org/10.1007/pl00012580.

COHEN, Edith et al. Computing classic closeness centrality, at scale. **Proceedings Of The Second Edition Of The Acm Conference On Online Social Networks - Cosn '14**, [s.l.], p.37-50, 2014. ACM Press. http://dx.doi.org/10.1145/2660460.2660465.

COSTA, L. da F. et al. Characterization of complex networks: A survey of measurements. **Advances In Physics**, [s.l.], v. 56, n. 1, p.167-242, jan. 2008. Informa UK Limited. http://dx.doi.org/10.1080/00018730601170527.

DANOWSKI, James A.; CEPELA, Noah T.. Automatic Mapping of Social Networks of Actors from Text Corpora: Time Series Analysis. **2009 International Conference On Advances In Social Network Analysis And Mining**, [s.l.], p.1-10, jul. 2009. IEEE. http://dx.doi.org/10.1109/asonam.2009.71.

DAS, Kousik; SAMANTA, Sovan; PAL, Madhumangal. Study on centrality measures in social networks: a survey. **Social Network Analysis And Mining**, [s.l.], v. 8, n. 1, p.1-10, 28 fev. 2018. Springer Nature. http://dx.doi.org/10.1007/s13278-018-0493-2.

DENNIS, J. E.; SCHNABEL, Robert B.. Numerical Methods for Unconstrained Optimization and Nonlinear Equations. **Classics In Applied Mathematics**, [s.l.], jan. 1996. 378 p. Society for Industrial and Applied Mathematics. http://dx.doi.org/10.1137/1.9781611971200.

DING, Jin; LU, Yong-zai. Control backbone: An index for quantifying a node′s importance for the network controllability. **Neurocomputing**, [s.l.], v. 153, p.309-318, abr. 2015. Elsevier BV. http://dx.doi.org/10.1016/j.neucom.2014.11.024.

EASLEY, David; KLEINBERG, Jon. **Networks, Crowds, and Markets:** Reasoning About a Highly Connected World. Cambridge, 2010. 744 p.

EPPSTEIN, David; WANG, Joseph. Fast Approximation of Centrality. **Graph Algorithms And Applications 5**, [s.l.], p.39-45, jun. 2006. WORLD SCIENTIFIC. http://dx.doi.org/10.1142/9789812773289_0004.

ERCSEY-RAVASZ, Mária et al. Range-limited centrality measures in complex networks. **Physical Review e**, [s.l.], v. 85, n. 6, p.1-10, 6 jun. 2012. American Physical Society (APS). http://dx.doi.org/10.1103/physreve.85.066103.

ERDŐS, P.; RÉNYI, A.. On Random Graphs I. **Publicationes Mathematicae**, v. 6, p.290-297, 1959.

ESTRADA, Ernesto; HIGHAM, Desmond J.; HATANO, Naomichi. Communicability betweenness in complex networks. **Physica A**: Statistical Mechanics and its Applications, [s.l.], v. 388, n. 5, p.764-774, mar. 2009. Elsevier BV. http://dx.doi.org/10.1016/j.physa.2008.11.011.

ESTRADA, Ernesto; RODRÍGUEZ-VELÁZQUEZ, Juan A.. Subgraph centrality in complex networks. **Physical Review e**, [s.l.], v. 71, n. 5, p.1-10, 6 maio 2005. American Physical Society (APS). http://dx.doi.org/10.1103/physreve.71.056103.

FACELI, Katti et al. Inteligência Artificial: Uma Abordagem de Aprendizado de Máquina. Rio de Janeiro, Rj: LTC., 2011. 378 p.

FIRE, Michael et al. Computationally efficient link prediction in a variety of social networks. **Acm Transactions On Intelligent Systems And Technology**, [s.l.], v. 5, n. 1, p.1-25, 1 dez. 2013. Association for Computing Machinery (ACM). http://dx.doi.org/10.1145/2542182.2542192.

FIRE, Michael et al. Link Prediction in Social Networks Using Computationally Efficient Topological Features. **2011 Ieee Third Int'l Conference On Privacy, Security, Risk And Trust And 2011 Ieee Third Int'l Conference On Social Computing**, [s.l.], p.1-10, out. 2011. IEEE. http://dx.doi.org/10.1109/passat/socialcom.2011.20.

FREEMAN, Linton C.. A Set of Measures of Centrality Based on Betweenness. **Sociometry**, [s.l.], v. 40, n. 1, p.35-41, mar. 1977. JSTOR. http://dx.doi.org/10.2307/3033543.

FREEMAN, Linton C.. Centrality in social networks conceptual clarification. **Social Networks**, [s.l.], v. 1, n. 3, p.215-239, 1978/79. Elsevier BV. http://dx.doi.org/10.1016/0378-8733(78)90021-7.

FREEMAN, Linton C.; BORGATTI, Stephen P.; WHITE, Douglas R.. Centrality in valued graphs: A measure of betweenness based on network flow. **Social Networks**, [s.l.], v. 13, n. 2, p.141-154, jun. 1991. Elsevier BV. http://dx.doi.org/10.1016/0378-8733(91)90017-n.

GAO, Song et al. Understanding Urban Traffic-Flow Characteristics: A Rethinking of Betweenness Centrality. **Environment And Planning B**: Planning and Design, [s.l.], v. 40, n. 1, p.135-153, jan. 2013. SAGE Publications. http://dx.doi.org/10.1068/b38141.

Garcez, Artur d'Avila et al. Neural-Symbolic Learning and Reasoning: Contributions and Challenges. **AAAI Spring Symposium on Knoledge Representation and Reasoning: Integrating Symbolic and Neural Approaches**, Stanford, p. 18-21, 2015.

GILL, Philip E.; MURRAY, Walter; WRIGHT, Margaret H.. **Practical Optimization**. Bingley, Emerald Publishing, 1981. 418p.

GOH, K.-i. et al. Betweenness centrality correlation in social networks. **Physical Review e**, [s.l.], v. 67, n. 1, p.1-10, 13 jan. 2003. American Physical Society (APS). http://dx.doi.org/10.1103/physreve.67.017101.

GRANDO, Felipe; GRANVILLE, Lisandro Z.; LAMB, Luis C.. Machine Learning in Network Centrality Measures: Tutorial and Outlook. **Acm Computing Surveys**, [s.l.], v. 51, n. 3, p.1-21, 2018. Association for Computing Machinery (ACM).

GRANDO, Felipe; LAMB, Luís C.. Computing vertex centrality measures in massive real networks with a neural learning model. **2018 International Joint Conference On Neural Networks (ijcnn)**, [s.l.], p.1-8, julho 2018a.

GRANDO, Felipe; LAMB, Luis C.. Estimating complex networks centrality via neural networks and machine learning. **2015 International Joint Conference On Neural Networks (ijcnn)**, [s.l.], p.1-8, jul. 2015. IEEE. http://dx.doi.org/10.1109/ijcnn.2015.7280334.

GRANDO, Felipe; LAMB, Luis C.. On approximating networks centrality measures via neural learning algorithms. **2016 International Joint Conference On Neural Networks (ijcnn)**, [s.l.], p.1-8, jul. 2016. IEEE. http://dx.doi.org/10.1109/ijcnn.2016.7727248.

GRANDO, Felipe; LAMB, Luís C.. On the effectiveness of the block two-level Erdős-Rényi generative network model. **2018 Ieee Symposium On Computers And Communications (iscc)**, [s.l.], p.1-6, junho 2018b.

GRANDO, Felipe; NOBLE, Diego; LAMB, Luis C.. An Analysis of Centrality Measures for Complex and Social Networks. **2016 Ieee Global Communications Conference (globecom)**, [s.l.], p.1-10, dez. 2016. IEEE. http://dx.doi.org/10.1109/glocom.2016.7841580.

HAGAN, M.t.; MENHAJ, M.b.. Training feedforward networks with the Marquardt algorithm. **Ieee Transactions On Neural Networks**, [s.l.], v. 5, n. 6, p.989-993, 1994. Institute of Electrical and Electronics Engineers (IEEE). http://dx.doi.org/10.1109/72.329697.

HAGAN, Martin T.; DEMUTH, Howard B.; BEALE, Mark H.; JESÚS, Orlando de. **Neural Network Design**. Boston, PWS Publishing, 1996. 800 p.

HAGE, Per; HARARY, Frank. Eccentricity and centrality in networks. **Social Networks**, [s.l.], v. 17, n. 1, p.57-63, jan. 1995. Elsevier BV. http://dx.doi.org/10.1016/0378-8733(94)00248-9.

HAYKIN, Simon. **Neural Networks:** A comprehensive foundation. 2. ed. Upper Saddle River, Nj: Prentice Hall, 1999. 842 p.

HEUVEL, Martijn P. van Den; SPORNS, Olaf. Network hubs in the human brain. **Trends In Cognitive Sciences**, [s.l.], v. 17, n. 12, p.683-696, dez. 2013. Elsevier BV. http://dx.doi.org/10.1016/j.tics.2013.09.012.

HUA, Guangying; SUN, Yingjie; HAUGHTON, Dominique. Network Analysis of US Air Transportation Network. **Data Mining For Social Network Data**, [s.l.], p.75-89, 2010. Springer US. http://dx.doi.org/10.1007/978-1-4419-6287-4_5.

JACKSON, Matthew O.; WATTS, Alison. The Evolution of Social and Economic Networks. **Journal Of Economic Theory**, [s.l.], v. 106, n. 2, p.265-295, out. 2002. Elsevier BV. http://dx.doi.org/10.1006/jeth.2001.2903.

JACOMY, Mathieu et al. ForceAtlas2, a Continuous Graph Layout Algorithm for Handy Network Visualization Designed for the Gephi Software. **Plos One**, [s.l.], v. 9, n. 6, p.1-10, 10 jun. 2014. Public Library of Science (PLoS). http://dx.doi.org/10.1371/journal.pone.0098679.

JAYASINGHE, Amila; SANO, Kazushi; NISHIUCHI, Hiroaki. EXPLAINING TRAFFIC FLOW PATTERNS USING CENTRALITY MEASURES. **International Journal For Traffic And Transport Engineering**, [s.l.], v. 5, n. 2, p.134-149, jun. 2015. City Net Scientific Research Center Ltd., Belgrade. http://dx.doi.org/10.7708/ijtte.2015.5(2).05.

JIANG, J.c.; YU, J.y.; LEI, J.s.. Finding influential agent groups in complex multiagent software systems based on citation network analyses. **Advances In Engineering Software**, [s.l.], v. 79, p.57-69, jan. 2015. Elsevier BV. http://dx.doi.org/10.1016/j.advengsoft.2014.09.002.

KAS, Miray et al. What if wireless routers were social? approaching wireless mesh networks from a social networks perspective. **Ieee Wireless Communications**, [s.l.], v. 19, n. 6, p.36-43, dez. 2012. Institute of Electrical and Electronics Engineers (IEEE). http://dx.doi.org/10.1109/mwc.2012.6393516.

KATZ, Leo. A new status index derived from sociometric analysis. **Psychometrika**, [s.l.], v. 18, n. 1, p.39-43, mar. 1953. Springer Nature. http://dx.doi.org/10.1007/bf02289026.

KAZIENKO, Przemyslaw; KAJDANOWICZ, Tomasz. Label-dependent node classification in the network. **Neurocomputing**, [s.l.], v. 75, n. 1, p.199-209, jan. 2012. Elsevier BV. http://dx.doi.org/10.1016/j.neucom.2011.04.047.

KLIMT, Bryan; YANG, Yiming. Introducing the Enron Corpus. **CEAS Aeronautical Journal**, [s.l.], v. 5, p.1-2, 2004. Springer.

KOLDA, Tamara G. et al. A Scalable Generative Graph Model with Community Structure. **Siam Journal On Scientific Computing**, [s.l.], v. 36, n. 5, p.424-452, jan. 2014. Society for Industrial & Applied Mathematics (SIAM). http://dx.doi.org/10.1137/130914218.

KONECT. **Network Dataset**, at http://konect.uni-klobenz.de/, 2016.

KÖNIG, Michael D.; TESSONE, Claudio J.; ZENOU, Yves. FROM ASSORTATIVE TO DISSORTATIVE NETWORKS: THE ROLE OF CAPACITY CONSTRAINTS. **Advances In Complex Systems**, [s.l.], v. 13, n. 04, p.483-499, ago. 2010. World Scientific Pub Co Pte Lt. http://dx.doi.org/10.1142/s0219525910002700.

KÖNIG, Michael D.; TESSONE, Claudio J.; ZENOU, Yves. Nestedness in networks: A theoretical model and some applications. **Theoretical Economics**, [s.l.], v. 9, n. 3, p.695-752, set. 2014. The Econometric Society. http://dx.doi.org/10.3982/te1348.

KUMAR, Ashok; MEHROTRA, Kishan G.; MOHAN, Chilukuri K.. Neural Networks for Fast Estimation of Social Network Centrality Measures. **Proceedings of the Fifth International Conference on Fuzzy and Neuro Computing (FANCCO – 2015)**, p.175-184, 2015.

LANDHERR, Andrea; FRIEDL, Bettina; HEIDEMANN, Julia. A Critical Review of Centrality Measures in Social Networks. **Business & Information Systems Engineering**, [s.l.], v. 2, n. 6, p.371-385, 20 out. 2010. Springer Nature. http://dx.doi.org/10.1007/s12599-010-0127-3.

LECUN, Yann; BENGIO, Yoshua; HINTON, Geoffrey. Deep learning. **Nature**, [s.l.], v. 521, n. 7553, p.436-444, maio 2015. Springer Nature. http://dx.doi.org/10.1038/nature14539.

LESKOVEC, Jure; CHAKRABARTI, Deepayan; KLEINBERG, Jon; FALOUTSOS, Christos; GHAHRAMANI, Zoubin. Kronecker Graphs: An Approach to Modeling Networks. **Journal of Machine Learning Research**, v. 11, p.985-1042, 2010.

LESKOVEC, Jure; KLEINBERG, Jon; FALOUTSOS, Christos. Graph evolution. **Acm Transactions On Knowledge Discovery From Data**, [s.l.], v. 1, n. 1, p.1-10, 1 mar. 2007. Association for Computing Machinery (ACM). http://dx.doi.org/10.1145/1217299.1217301.

LESKOVEC, Jure; KLEINBERG, Jon; FALOUTSOS, Christos. Graphs over time. **Proceeding Of The Eleventh Acm Sigkdd International Conference On Knowledge Discovery In Data Mining - Kdd '05**, [s.l.], p.177-187, 2005. ACM Press. http://dx.doi.org/10.1145/1081870.1081893.

LESKOVEC, Jure; LANG, Kevin J.; DASGUPTA, Anirban; MAHONEY, Michael W.. Community Structure in Large Networks: Natural Cluster Sizes and the Absence of Large Well-Defined Clsuters. **Internet Mathematics**, v. 6, n. 1, p.29-123, 2009.

LESKOVEC, Jure; MCAULEY, Julian J.. Learning to Discover Social Circles in Ego Networks. **Advances in Neural Information Processing Systems 25 (NIPS 2012)**, p.1-10, 2012.

LI, Cong et al. Correlation between centrality metrics and their application to the opinion model. **The European Physical Journal B**, [s.l.], v. 88, n. 3, p.1-10, mar. 2015. Springer Nature. http://dx.doi.org/10.1140/epjb/e2015-50671-y.

LI, Xujun et al. Identifying social influence in complex networks: A novel conductance eigenvector centrality model. **Neurocomputing**, [s.l.], v. 210, p.141-154, out. 2016. Elsevier BV. http://dx.doi.org/10.1016/j.neucom.2015.11.123.

LIBEN-NOWELL, David; KLEINBERG, Jon. **The link-prediction problem for social networks. Journal Of The American Society For Information Science And Technology**, [s.l.], v. 58, n. 7, p.1019-1031, 2007. Wiley. http://dx.doi.org/10.1002/asi.20591.

LOUATI, Amine; HADDAD, Joyce El; PINSON, Suzanne. A Multilevel Agent-Based Approach for Trustworthy Service Selection in Social Networks. **2014 Ieee/wic/acm International Joint Conferences On Web Intelligence (wi) And Intelligent Agent Technologies (iat)**, [s.l.], p.1-10, ago. 2014. IEEE. http://dx.doi.org/10.1109/wi-iat.2014.170.

LUGER, George F.. **Artificial Intelligence:** Structures and Strategies for Complex Problem Solving. 4. ed. Harlow, England: Pearson Education, 2002. 856 p.

MACCARI, Leonardo; LOCIGNO, Renato. Pop-routing: Centrality-based tuning of control messages for faster route convergence. **Ieee Infocom 2016 - The 35th Annual Ieee International Conference On Computer Communications**, [s.l.], p.1-10, abr. 2016. IEEE. http://dx.doi.org/10.1109/infocom.2016.7524407.

MACCARI, Leonardo; LOCIGNO, Renato. Waterwall: a cooperative, distributed firewall for wireless mesh networks. **Eurasip Journal On Wireless Communications And Networking**, [s.l.], v. 2013, n. 1, p.1-10, 8 set. 2013. Springer Nature. http://dx.doi.org/10.1186/1687-1499-2013-225.

MACCARI, Leonardo; NGUYEN, Quynh; LOCIGNO, Renato. On the Computation of Centrality Metrics for Network Security in Mesh Networks. **2016 Ieee Global Communications Conference (globecom)**, [s.l.], p.1-10, dez. 2016. IEEE. http://dx.doi.org/10.1109/glocom.2016.7842049.

MACKAY, David J. C.. Bayesian Interpolation. **Neural Computation**, [s.l.], v. 4, n. 3, p.415-447, maio 1992. MIT Press - Journals. http://dx.doi.org/10.1162/neco.1992.4.3.415.

MARCHANT, James; GRIFFITHS, Nathan. Manipulating Conventions in a Particle-Based Topology. **Lecture Notes In Computer Science**, [s.l.], p.242-261, 2016. Springer International Publishing. http://dx.doi.org/10.1007/978-3-319-42691-4_14.

MICHALAK, Tomasz Pawel et al. Efficient Computation of the Shapley Value for Game-Theoretic Network Centrality. **Journal Of Artificial Intelligence Research**, [s.l.], v. 46, p.1-10, 2013. http://dx.doi.org/10.1613/jair.3806.

MITCHELL, Tom M.. **Machine Learning**. Mcgraw-hill Science/engineering/math, 1997. 432 p.

MITZENMACHER, Michael. A Brief History of Generative Models for Power Law and Lognormal Distributions. Internet Mathematics, [s.l.], v. 1, n. 2, p.226-251, jan. 2004. **Internet Mathematics**. http://dx.doi.org/10.1080/15427951.2004.10129088.

MØLLER, Martin Fodslette. A scaled conjugate gradient algorithm for fast supervised learning. **Neural Networks**, [s.l.], v. 6, n. 4, p.525-533, jan. 1993. Elsevier BV. http://dx.doi.org/10.1016/s0893-6080(05)80056-5.

MORADI, Parham; SHIRI, Mohammad Ebrahim; ENTEZARI, Negin. Automatic Skill Acquisition in Reinforcement Learning Agents Using Connection Bridge Centrality. **Communication And Networking**, [s.l.], p.51-62, 2010. Springer Berlin Heidelberg. http://dx.doi.org/10.1007/978-3-642-17604-3_6.

NEWMAN, M. E. J.; GIRVAN, M.. Finding and evaluating community structure in networks. **Physical Review e**, [s.l.], v. 69, n. 2, p.1-10, 26 fev. 2004. American Physical Society (APS). http://dx.doi.org/10.1103/physreve.69.026113.

NEWMAN, M. E. J.; PARK, Juyong. Why social networks are different from other types of networks. **Physical Review e**, [s.l.], v. 68, n. 3, p.1-10, 22 set. 2003. American Physical Society (APS). http://dx.doi.org/10.1103/physreve.68.036122.

NEWMAN, M.e. J.. A measure of betweenness centrality based on random walks. **Social Networks**, [s.l.], v. 27, n. 1, p.39-54, jan. 2005. Elsevier BV. http://dx.doi.org/10.1016/j.socnet.2004.11.009.

NIEMINEN, Juhani. On the centrality in a graph. **Scandinavian Journal Of Psychology**, [s.l.], v. 15, n. 1, p.332-336, set. 1974. Wiley. http://dx.doi.org/10.1111/j.1467-9450.1974.tb00598.x.

NOBLE, Diego Vrague et al. The Impact of Centrality on Individual and Collective Performance in Social Problem-Solving Systems. **Proceedings Of The 2015 On Genetic And Evolutionary Computation Conference - Gecco '15**, [s.l.], p.1-8, 2015. ACM Press. http://dx.doi.org/10.1145/2739480.2754679.

NOH, Jae Dong; RIEGER, Heiko. Random Walks on Complex Networks. **Physical Review Letters**, [s.l.], v. 92, n. 11, p.1-10, 18 mar. 2004. American Physical Society (APS). http://dx.doi.org/10.1103/physrevlett.92.118701.

PAGE, Lawrence; BRIN, Sergey; MOTWANI, Rajeev; WINOGRAD, Terry. The PageRank Citation Ranking: Bringing Order to the Web. **Technical Report**, Stanford InfoLab, 1999. 8 p.

PANTAZOPOULOS, Panagiotis; KARALIOPOULOS, Merkouris; STAVRAKAKIS, Ioannis. Distributed Placement of Autonomic Internet Services. **Ieee Transactions On Parallel And Distributed Systems**, [s.l.], v. 25, n. 7, p.1702-1712, jul. 2014. Institute of Electrical and Electronics Engineers (IEEE). http://dx.doi.org/10.1109/tpds.2013.186.

POWELL, M. J. D.. Restart procedures for the conjugate gradient method. **Mathematical Programming**, [s.l.], v. 12, n. 1, p.241-254, dez. 1977. Springer Nature. http://dx.doi.org/10.1007/bf01593790.

RAD, Ali Ajdari; HASLER, Martin; MORADI, Parham. Automatic skill acquisition in Reinforcement Learning using connection graph stability centrality. **Proceedings Of 2010 Ieee International Symposium On Circuits And Systems**, [s.l.], p.1-10, maio 2010. IEEE. http://dx.doi.org/10.1109/iscas.2010.5537485.

RICHARDS, William; SEARY, Andrew. Eigen analysis of Networks. **Structural Nalysis and Journal of Social Structure**, Burnaby, Canada, 2000. 15 p.

RIEDMILLER, M.; BRAUN, H.. A direct adaptive method for faster backpropagation learning: the RPROP algorithm. **Ieee International Conference On Neural Networks**, [s.l.], p.1-10, 1993. IEEE. http://dx.doi.org/10.1109/icnn.1993.298623.

RODRÍGUEZ, J. A.; ESTRADA, E.; GUTIÉRREZ, A.. Functional centrality in graphs. **Linear And Multilinear Algebra**, [s.l.], v. 55, n. 3, p.293-302, maio 2007. Informa UK Limited. http://dx.doi.org/10.1080/03081080601002221.

RUBINOV, Mikail; SPORNS, Olaf. Complex network measures of brain connectivity: Uses and interpretations. **Neuroimage**, [s.l.], v. 52, n. 3, p.1059-1069, set. 2010. Elsevier BV. http://dx.doi.org/10.1016/j.neuroimage.2009.10.003.

SABIDUSSI, Gert. The centrality index of a graph. **Psychometrika**, [s.l.], v. 31, n. 4, p.581-603, dez. 1966. Springer Nature. http://dx.doi.org/10.1007/bf02289527.

SCALES, L. E.. **Introduction to Non-Linear Optimization**. New York, Springler - Verlag, 1985. 256 p.

SESHADHRI, C.; KOLDA, Tamara G.; PINAR, Ali. Community structure and scale-free collections of Erdős-Rényi graphs. **Physical Review e**, [s.l.], v. 85, n. 5, p.1-10, 10 maio 2012. American Physical Society (APS). http://dx.doi.org/10.1103/physreve.85.056109.

SHAW, Marvin E.. Group Structure and the Behavior of Individuals in Small Groups. **The Journal Of Psychology**, [s.l.], v. 38, n. 1, p.139-149, jul. 1954. Informa UK Limited. http://dx.doi.org/10.1080/00223980.1954.9712925.

SMOLA, Alex; VISHWANATHAN, S. V. N.. **Introduction to Machine Learning**. Cambridge, Uk: Cambridge University, 2008. 234 p.

SOUNDARAJAN, Sucheta; ELIASSI-RAD, Tina; GALLAGHER, Brian. A Guide to Selecting a Network Similarity Method. **Proceedings Of The 2014 Siam International Conference On Data Mining**, [s.l.], p.1037-1045, 28 abr. 2014. Society for Industrial and Applied Mathematics. http://dx.doi.org/10.1137/1.9781611973440.118.

STEPHENSON, Karen; ZELEN, Marvin. Rethinking centrality: Methods and examples. **Social Networks**, [s.l.], v. 11, n. 1, p.1-37, mar. 1989. Elsevier BV. http://dx.doi.org/10.1016/0378-8733(89)90016-6.

ŠUBELJ, L.; BAJEC, M.. Robust network community detection using balanced propagation. **The European Physical Journal B**, [s.l.], v. 81, n. 3, p.353-362, 4 maio 2011. Springer Nature. http://dx.doi.org/10.1140/epjb/e2011-10979-2.

TRAN, Tien-dzung; KWON, Yung-keun. Hierarchical closeness efficiently predicts disease genes in a directed signaling network. **Computational Biology And Chemistry**, [s.l.], v. 53, p.191-197, dez. 2014. Elsevier BV. http://dx.doi.org/10.1016/j.compbiolchem.2014.08.023.

VALENTE, T. W.; CORONGES, K.; LAKON, C.; COSTENBADER, E.. How correlated are network centrality measures? **Connect (Tor)**. v. 28, n. 1, p.16-26, 2008.

VÁZQUEZ-RODAS, Andrés; LLOPIS, Luis J. de La Cruz. A centrality-based topology control protocol for wireless mesh networks. **Ad Hoc Networks**, [s.l.], v. 24, p.34-54, jan. 2015. Elsevier BV. http://dx.doi.org/10.1016/j.adhoc.2014.07.026.

WATTS, Duncan J.; STROGATZ, Steven H.. Collective dynamics of 'small-world' networks. **Nature**, [s.l.], v. 393, n. 6684, p.440-442, jun. 1998. Springer Nature. http://dx.doi.org/10.1038/30918.

XIONG, Wei et al. Active learning for protein function prediction in protein–protein interaction networks. **Neurocomputing**, [s.l.], v. 145, p.44-52, dez. 2014. Elsevier BV. http://dx.doi.org/10.1016/j.neucom.2014.05.075.

XU, Yang; HU, Xuemei; Li, Dong; YANG, Mengjun. Using complex network effects for communication decisions in large multi-robot teams. **AAMAS**, p.1-10, 2014.

YAN, Erjia; DING, Ying. Applying centrality measures to impact analysis: A coauthorship network analysis. **Journal Of The American Society For Information Science And Technology**, [s.l.], v. 60, n. 10, p.2107-2118, out. 2009. Wiley. http://dx.doi.org/10.1002/asi.21128.

YANG, Jaewon; LESKOVEC, Jure. Defining and evaluating network communities based on ground-truth. Proceedings Of The Acm Sigkdd Workshop On Mining Data Semantics - Mds '12, [s.l.], p.1-10, 2012. ACM Press. http://dx.doi.org/10.1145/2350190.2350193.

YU, Shui et al. Networking for Big Data: A Survey. **Ieee Communications Surveys & Tutorials**, [s.l.], v. 19, n. 1, p.531-549, 2017. Institute of Electrical and Electronics Engineers (IEEE). http://dx.doi.org/10.1109/comst.2016.2610963.

ZAFARANI, R.; LIU, H.. **Social Computing Data Repository at ASU**, Arizona State University, School of Computing, Informatics and Decision Systems Engineering, 2009.

ZAK, Blazej; ZBIEG, Anita. Heuristic for Network Coverage Optimization Applied in Finding Organizational Change Agents. **2014 European Network Intelligence Conference**, [s.l.], p.1-10, set. 2014. IEEE. http://dx.doi.org/10.1109/enic.2014.27.

ZHAO, P. X.; ZHAO, S. M.. UNDERSTANDING URBAN TRAFFIC FLOW CHARACTERISTICS FROM THE NETWORK CENTRALITY PERSPECTIVE AT DIFFERENT GRANULARITIES. **Isprs - International Archives Of The Photogrammetry, Remote Sensing And Spatial Information Sciences**, [s.l.], v. -2, p.263-268, 7 jun. 2016. Copernicus GmbH. http://dx.doi.org/10.5194/isprsarchives-xli-b2-263-2016.

ZILBERMAN, Polina; PUZIS, Rami; ELOVICI, Yuval. On Network Footprint of Traffic Inspection and Filtering at Global Scrubbing Centers. **Ieee Transactions On Dependable And Secure Computing**, [s.l.], v. 14, n. 5, p.521-534, 1 set. 2017. Institute of Electrical and Electronics Engineers (IEEE). http://dx.doi.org/10.1109/tdsc.2015.2494039.