Ben-Gurion University of the Negev
The Faculty of Engineering Science
**The Department of Software and Information Systems Engineering**

# Learning Centrality By Learning To Route

Thesis submitted in partial fulfillment of the requirements
for the Master of Sciences degree

**Submitted by**

Liav Bachar

Department Of Information Systems Engineering

Ben-Gurion University of the Negev

Tel: +972-525213367

E-mail: bacharliav@gmail.com

**April 2022**

Ben-Gurion University of the Negev
The Faculty of Natural Sciences
The Department of **Computer Science**

# Learning Centrality By Learning To Route

Thesis submitted in partial fulfillment of the requirements
for the Master of Sciences degree

**By Liav Bachar**

Under the supervision of **Mr. Aviad Elyashar and Dr. Rami Puzis**

| | | | |
|---|---|---|---|
| Signature of student: | [illegible] | Date: | 17/04/2022 |
| Signature of supervisor: | | Date: | |
| Signature of chairperson of the committee for graduate studies: | | Date: | |

**April 2022**

# Abstract

In graph theory and network analysis, centrality measures are properties of nodes that rank their importance according to a given task. Examples of these tasks are controlling and monitoring traffic in complex networks, finding influential users in social networks, and detecting bots on the internet. To achieve a meaningful importance ranking it is important to select a suitable centrality measure for the given task. Although researchers invented many centrality measures over the years, from time to time, a new task that was not covered yet by centrality measures shows up and requires the invention of a tailored centrality measure.

Developing a tailor-made centrality measure for a given task requires domain and network analysis expertise, as well as time and effort. Automatically learning arbitrary centrality measures provided ground truth node scores is an important research direction.

In this thesis, we propose a generic deep learning architecture for centrality learning that relies on the insight that arbitrary centrality measures can be computed using Routing Betweenness Centrality (RBC) and our new differentiable implementation of RBC. The proposed Learned Routing Centrality (LRC) architecture optimizes the routing f unction of RBC to fit the ground truth scores. Results show that LRC can learn multiple types of centrality indices more accurately than state-of-the-art.

***Keywords***— Complex Networks, Centrality Measures, Random Geometric Graphs, Neural Networks, Eigenvector, Node Embedding, Routing Policy

# Acknowledgements

I would like to express my sincere gratitude to my supervisors, **Dr. Rami Puzis and Mr. Aviad Elyashar** for their continuous support of my M.Sc. studies and research —and especially for their patience, motivation, enthusiasm, and immense knowledge. Their guidance has helped me throughout my research and in the construction of this thesis.

# Contents

# List of Figures

# List of Tables

# 1 Introduction

Network science is ubiquitous with application domains in many disciplines including biology [1], electrical engineering [2], economics, public health, and neuroscience [3]. Network analysis provides the capacity to estimate complex patterns of relationships and the network structure can be analyzed to reveal core features of the network.

Centrality computation is one of the most important and commonly used tools in the analysis of complex networks. Centrality measures are properties of nodes that correspond to the nodes' importance for different tasks. For example, nodes with high betweenness centrality can be used to monitor network flows [4, 5], nodes with high closeness can easily reach all other network participants, they can be used to disseminate information [6], etc. Thus, the same node may have to be ranked differently by different centrality measures.

Over the years, many different centrality measures were proposed [7]. Researchers continue inventing new centrality measures and node properties from time to time to fit a particular task that was not properly covered yet [8, 9]. Formulating new structural node properties, including centrality measures, for a task at hand requires domain expertise, network analysis expertise, and of course time and effort. In recent years, machine learning (ML) and deep learning (DL) techniques on graphs, such as Graph Convolutional Networks (GCNs) [10]) or [11], were developed to learn from the structure and properties of nodes. In this study, we focus on learning new centrality measures of nodes from the graph structure.

Standard ML techniques previously proposed for learning centrality measures rely on some centrality measures as features to compute a new centrality measure [12–14]. They show good correlations with the ground truth centrality. However, these results can be explained by the natural correlation between the target centrality and the one used as a feature. Such models inherit the pros and cons of the centrality measures used as features and may not predict well entirely new centrality. Centrality learning by Graph Neural Networks (GNN) [15] does not rely on pre-computed centrality measures. Unfortunately, state-of-the-art approaches, such as [15] use different DL architectures for learning closeness and betweenness measures, falling back to relying on human expertise.

This thesis proposes a novel technique for learning heterogeneous centrality measures on graphs using the same generic architecture. Our approach relies on the concept of Routing Betweenness Centrality (RBC) [16] that generalizes other betweenness measures, such as the Shortest Path Betweenness (SPBC) [17], and Traffic Load Centrality (Load) [18], by considering network flows created by arbitrary loop-free routing strategies. Instead of learning the centrality measure directly, we use DL techniques to learn the RBC routing function.

## 1.1 Contributions

This thesis was carried out to develop a novel technique for learning centrality measures for a given task automatically. Defining new centrality measures for a given task by researcher requires domain and network analysis expertise, time, and effort. Specifically, this thesis offers the following contributions:

1. We develop the Learned Routing Centrality (LRC) architecture,[1] that learns a routing function from node geometric embedding to compute arbitrary target centrality measures. The proposed LRC technique produces higher correlations with the target centrality measure than state-of-the-art ML techniques.

2. We propose a new differentiable algebraic computation of RBC that facilitates its incorporation within deep learning architectures.

## 1.2 Publication

This thesis was a product of a fruitful collaboration between the author of the dissertation along with Aviad Elyashar and Dr. Rami Puzis.

[1] https://github.com/Liavbapp/LRC

It is worth mentioning that the work presented herein consists of research study that have been published in Complex Networks and their Applications international conference [19].

## 1.3 Organization

The rest of this thesis is structured as follows: Chapter 2 provides background on centrality measures, correlation measurements, node embedding, Random Geometric Graphs (RGG), and the Routing Betweenness Centrality (RBC) algorithm. Chapter 3 presents related work, whereas chapter 4 discusses the hypothesis and the LRC architecture. Chapter 5 shows the experiments conducted on arbitrary graphs and geometric graphs and then presents the results and discussion. Finally, chapter 6 discusses the conclusions, limitations, and future work.

# 2 Background

This thesis is about learning centrality measures on graphs, and it relates to the past work on Routing Betweenness Centrality (RBC). The first part of this section will introduce the following centrality measures: *Degree*, *Eigenvector*, *Closeness*, *Shortest Path Betweennes*, and *Load*. Later, in the experiments section, we will demonstrate the architecture performance on these centrality measures. The second part will discuss correlation measurements, specifically Pearson, Spearman, and Kendall. In this thesis, we use correlation measurements to optimize the loss function of the LRC architecture because the correlation measure goal is to predict the correct rank ordering of a given centrality measure for all nodes, not the resulting absolute rank value. The third section will discuss node embedding and will expand on GLEE [20] node embedding technique which we select as the embedding method to learn centrality measures on arbitrary graphs. The fourth section will present Random Geometric Graphs (RGG), the type of graphs on which LRC learned the centrality measures successfully. The last part will discuss the past work RBC and how this thesis utilizes it to develop the LRC architecture.

## 2.1 Node Centrality Measures

Centrality is a fundamental concept in network analysis. Bavelas [21, 22] first used centrality to explain the differential performance of communication networks, Shimbel [23] introduced a centrality measure based on shortest paths, to measure communication flows. Centrality also has been used to detect anomalies in networks [24], identification of influential nodes in social networks [25] and monitor traffic in transportation networks [26]. In this subsection, we will describe some important centrality measures that are considered in this thesis. In this context, network is a graph $G = (V, E)$ where $V$ is the set of nodes and $E$ is the set of edges in the graph, and

$A$ is adjacency matrix that represent the connectivity of the graph, where $a_{ij} \in A$ is the weight of the edges that connecting node $i$ to node $j$. for unweighted graphs, $a_{ij} = 1$ if there exist an edge between node $i$ to node $j$, otherwise $a_{ij} = 0$. Centrality measure gives a score for each node in the graph based on how this node is influential according to the measure.

### 2.1.1 Degree Centrality

The degree centrality of a vertex $i$ is the number of edges connected to it.

$$d_i = \sum_{j=1}^{n} a(i,j)$$

### 2.1.2 Eigenvector Centrality

The eigenvector centrality is a measure of the influence of a node in a network. Relative scores are assigned to all nodes in the network based on the concept that connections to high-scoring nodes contribute more to the score of the node in question than equal connections to low-scoring nodes. A high eigenvector score means that a node is connected to many nodes who themselves have high scores. The relative centrality, $x$, score of vertex $i$ can be defined as:

$$x_i = \frac{1}{\lambda} \sum_{t \in M(i)} x_t = \frac{1}{\lambda} \sum_{t \in G} a_{i,t} x_t$$

Where $M(i)$ is a set of the neighbors of $i$ and $\lambda$ is constant. With small rearrangement, this can be rewritten in vector notation as the eigenvector equation

$$Ax = \lambda x$$

In general, there will be many different eigenvalues $\lambda$ for which a non-zero eigenvector solution exists. However, the additional requirement that all the entries in the eigenvector be non-negative implies that only the greatest eigenvalue results in the desired centrality measure [27].

### 2.1.3 Closeness Centrality

The closeness centrality of a node represents how close the node to all other nodes in the graph, and is computed as the sum of the lengths of all the shortest path between the node and all other nodes in the graph. The more the node is close to all other nodes in the graph, the higher its closeness centrality. The formal definition of closeness centrality for node *i* is given by:

$$c_i = \frac{N-1}{\sum_{j \neq i \in V} \delta(i,j)}$$

where $N = |V|$ and $\delta(i,j)$ is the length of the shortest path between node $i$ to node $j$. As can be seen from the equation, the more node is close to other nodes, the sum in the denominator is smaller, and thus the value of $c_i$ is bigger.

### 2.1.4 Shortest Path Betweenness Centrality

The shortest path betweenness centrality (SPBC) was introduced in social sciences to measure the potential influence of an individual over the information flow in a social network [17, 28]. It represents how many shortest paths pass through a node. The more shortest the path pass through a node, the higher its SPBC centrality. The formal definition of SPBC of node *i* is given by:

$$b_i = \sum_{s \neq i \neq t} \frac{\sigma_{st}(i)}{\sigma_{st}}$$

where $\sigma_{st}$ is the total number of shortest paths from node $s$ to node $t$ and $\sigma_{st(i)}$ is the number of those paths that pass thorough $i$. As can be seen from the equation, the more paths pass through node $i$ the numerator is higher, and the fraction is closer to 1.

### 2.1.5 Traffic Load Centrality

Load Centrality (LC) as defined by Goh et al [29] is a variant of SPBC that also assumes that traffic flows over shortest paths, but uses a different routing mechanism, where every vertex forewords the packet to neighbor chosen randomly out of the neighbors that are closest to the target.

## 2.2 Correlation measurements

Correlation is a statistical measure that expresses the extent to which two variables are linearly related (meaning they change together at a constant rate). It's a common tool for describing simple relationships without making a statement about cause and effect. In this thesis, Pearson correlation used by the LRC architecture to compute loss while training. Moreover, both Kendall and Spearman correlations are used to evaluate the trained model performance.

### 2.2.1 Kendall Coefficient

The Kendall $\tau$-b rank correlation coefficient [30] between two variables ranges from -1 to 1. If the observations have a similar rank, then the correlation score will be close to 1, and in the case of identical observations, the score will be 1. The more observations are reversed, the more the correlation score will be closer to -1, and in the case of fully reversed observations, the score will be -1. If there is no correlation between the observations the correlation score will be 0. For our purpose, the two variables are a vector of node's ranking produced by LRC and the actual vector of node's ranking. The goal is that Kendall $\tau$-b test on these two variables to get a score close to 1, indicating there is a high correlation between the learned ranks to actual ranks.

### 2.2.2 Spearman's rank correlation coefficient

Spearman's rank correlation coefficient assesses how well the relationship between two variables can be described using a monotonic function and appropriate for both continuous and discrete ordinal variables. The Spearman correlation between two variables ranges from -1 to 1. Similar to Kendall $\tau$-b correlation score close to 1 indicates a high correlation, a score close to -1 indicates reversed correlation, and zero scores indicate no correlation. Our goal is to yield a correlation score close to 1. Although the similarity to Kendall $\tau$-b, if there are any ties in the data, irrespective of whether the percentage of ties is small or large, Spearman's measure returns values closer to the desired coverage rates, whereas Kendall's results differ more and more from the desired level as the number of ties increases, especially for large correlation values [31].

### 2.2.3 Pearson correlation coefficient

Pearson correlation is a measure of linear correlation between two variables. It is the ratio between the covariance of two variables and the product of their standard deviations. The difference between Spearman correlation to Pearson correlation is that Pearson works with a linear relationship between the two variables whereas Spearman works with a monotonic relationship as well, and Pearson works with raw data values of the variables whereas Spearman works with rank-ordered variables. Similar to Kendall $\tau$-b correlation and Pearson correlation score close to 1 indicates a high correlation, a score close to -1 indicates reversed correlation, and zero scores indicate no correlation.

## 2.3 Node embedding

As there is an increasing amount of graph data, ranging from social networks to various information networks, an important question that arises is how to represent a graph for analytics[32]. Node Embedding is the state-of-the-art nodes representation framework, which aims to project a node into a low-dimensional representation. This low-dimensional representation is then used for various downstream tasks [33, 34]. Some node embedding techniques are based on the spectral properties of the nodes. The idea of these techniques is that the embedding of a graph must respect node similarity - similar nodes must have embeddings that are close to one another. Other techniques dispose of the distance-minimization assumption, these techniques try to find embedding of the graph with geometric properties instead of spectral ones by leveraging the so-called simplex geometric of a graph. In this thesis, the GLEE node embedding method is used by the LRC architecture when training a model on arbitrary graphs.

### 2.3.1 GLEE

Geometric Laplacian Eigenmap Embedding (GLEE) [20] is a node embedding method that uses the Laplacian matrix to find an embedding with geometric properties instead of spectral ones, by leveraging the so-called simplex geometry of a graph. GLEE has deep connections with the so-called simplex geometry of the Laplacian [35, 36]. Fiedler [36] first made this observation, which highlights the bijective correspondence between the Laplacian matrix of an undirected, weighted graph and a geometric object known as a simplex. Using this relationship, GLEE finds graph embedding such that the representations $s_i$,$s_j$ of two non-adjacent nodes i and j are always orthogonal, $s_i \;\; s_j$ = 0, thus achieving a geometric encoding of adjacency.

## 2.4 Random Geometric Graph

A Geometric Graph (GG) is a graph drawn in the plane so that the nodes are represented by points in general position and the edges are represented by straight line segments connecting the corresponding points. Random Geometric Graph (RGG) is a GG where the nodes are randomly placed in the space and connecting two nodes by a link if their distance (according to some metric) is in a given range, e.g. smaller than a certain neighborhood radius [37]. The simplest case for node distribution is to scatter them uniformly in the embedding space such example consider a graph $G$ with $N = 200$ nodes and radius $R = 0.1$, where the nodes distributed over a square region of size 1 $x$ 1 is given in fig 2.1. RGG networks are characterized by their density which is defined by the number of nodes over the area of the network and by the connectivity radius size. The larger the radius is, the more neighbors each node will have, and thus, the more clustered the network will be.

## 2.5 Routing Betweenness Centrality

In 2010, Dolev et al. [16] proposed the Routing Betweenness Centrality (RBC) measure that generalizes previously well-known betweenness centrality measures, such as Shortest Path Betweenness Centrality (SPBC), Flow Betweenness Centrality (FBC), and Load Centrality (LC). These centrality measures include a fixed communication model that does not fully match routing strategies used in communication networks (e.g., the Internet). RBC is a flexible and realistic measure that accommodates a broad class of routing strategies. It measures the extent to which nodes or groups of nodes are exposed to the traffic given any loop-free routing strategy.

Let $G = (V, E)$ be a simple unweighted undirected graph with $|V| = n$ nodes and $|E| = m$ edges. Dolev et al. defined a routing policy $R : V^4 \rightarrow \mathbb{R}$ as the probability $p = R(s, t, u, v)$ that the node $u$ with forward to $v$ the packet sent from $s$ to $t$. A traffic matrix $T(s, t)$ is the number of packets sent from $s$ to $t$. Multiple RBC algorithms proposed by the authors receive $G$,

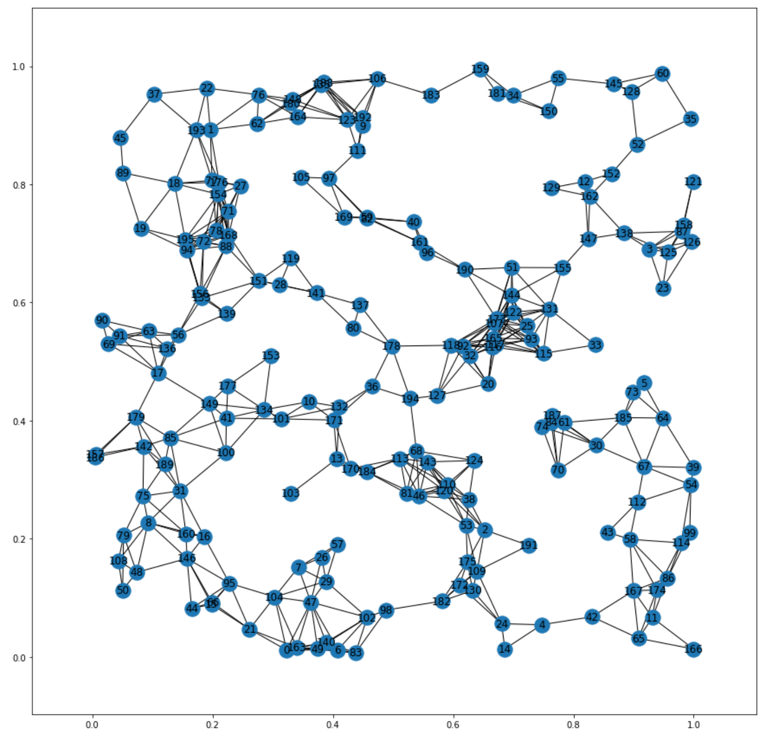

Figure 2.1: random geometric graph

$R$, and $T$ as an input and return the RBC of all nodes. RBC is equivalent to different betweenness measures under other conditions. For example, if $R(s,t,u,v)$ is the inverse of the number of $v$'s neighbors on the shortest path to $t$ and $T(s,t) = \left\{ \begin{smallmatrix} 0 & s=t \\ 1 & else \end{smallmatrix} \right.$, then RBC is equal to Newman's Load centrality.

We assume the above $T(s,t)$ throughout this thesis.

Following the common betweenness centrality notation, Dolev et al. denoted the probability of a packet passing through $v$ on a way from $s$ to $t$ as $\delta_{st}(v)$. Therefore, a RBC vector can be computed as a sum of $\delta_{st}$ over all $s,t \in V$ pairs.

$$RBC = \sum_{s,t \in V} \delta_{st} \tag{2.1}$$

Even though RBC can mimic multiple centrality measures, defining an appropriate routing policy is a challenging task that requires a deep understanding of the target centrality measure. The architecture developed in this thesis addresses this problem by automatically learning the routing function using deep learning techniques.

The authors discussed combinatorial computation of RBC similar to Brandes [38]. The running time complexity of the most efficient RBC algorithm is $O(nm)$ when the routing policy does not depend on the packet source. When $R$ depends on all four inputs, the complexity of computing RBC increases to $O(n^2m)$. RBC is useful for predicting the effectiveness and the cost of passive network monitoring. Although RBC is described in terms of communication networks, it can easily be applied for a wide variety of network-related tasks. In this thesis, we present an alternative differentiable algebraic computation of RBC with $O(n^4)$ running time complexity.

### 2.5.1 Example of using RBC to compute In-Degree of graph

Given the directed graph in Fig 2.2 (left), we would like to compute its in-degree by using the RBC algorithm. The first step is to define the routing policy for this graph. The routing policy is defined as the following:

$R(s,t,u,v) = \left\{ \begin{smallmatrix} 1 & u=s,t=v \\ 0 & else \end{smallmatrix} \right.$

Fig 2.2 (right) is a visual representation of the routing policy.

After defining the routing policy, we use equation 2.1 to compute the RBC of the graph. Note that if there isn't an edge between two nodes $s$ and $t$, then

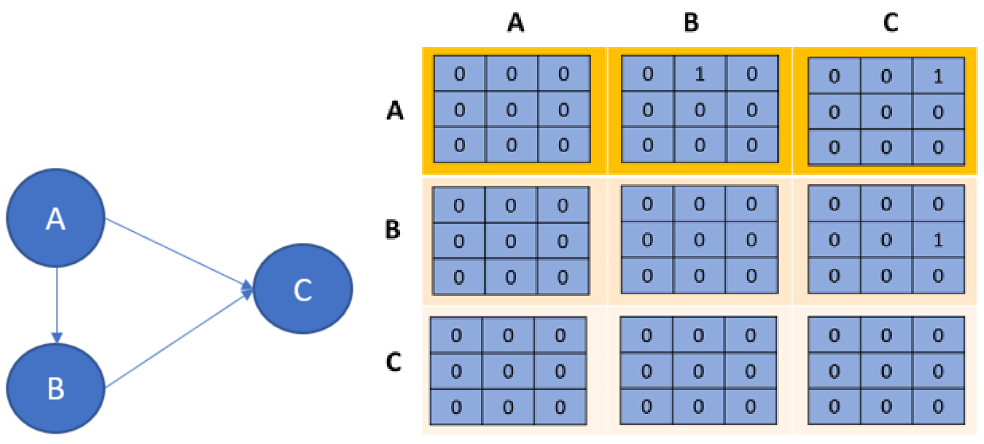

Figure 2.2: Directed Graph (left) and its routing policy(right), the routing policy is defined to compute the in-degree of the graph with the RBC algorithm.

$\forall_{u,v \in VxV}\ R(s,t,u,v) = 0$

In such case, the $delta_{st}$ vector will be vector of zeros as in Fig 2.3 (up). Otherwise if there is an edge between $s$ and $t$, then this edge contribute 1 to the $delta_{st}$ vector in the $t$ index as in Fig 2.3 (down). The final step is to sum all the $delta_{st}$ vectors which result in the In-Degree of the graph as illustrated in Fig 2.4.

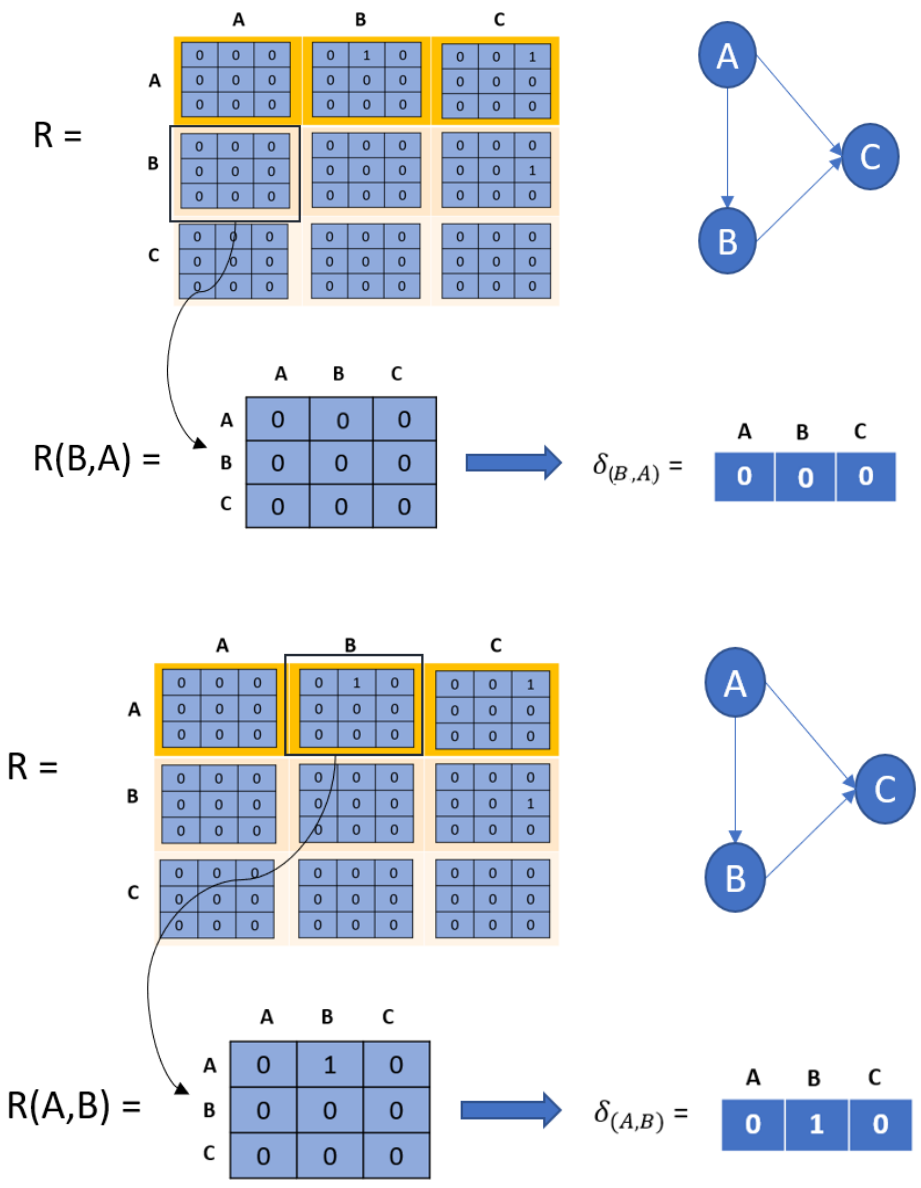

Figure 2.3: Computing the $delta_{st}$ vectors of the graph, the upper figure show computation of $delta_{BA}$ vector where there isn't an edge between $B$ and $A$ in the graph and thus the resulting $delta_{BA}$ vector is zeros vector. The bottom figure shows the computation of $delta_{AB}$ vector where there is an edge between $A$ and $B$ in the graph. The edge contributes 1 to the $delta_{AB}$ vector at the $B$ index.

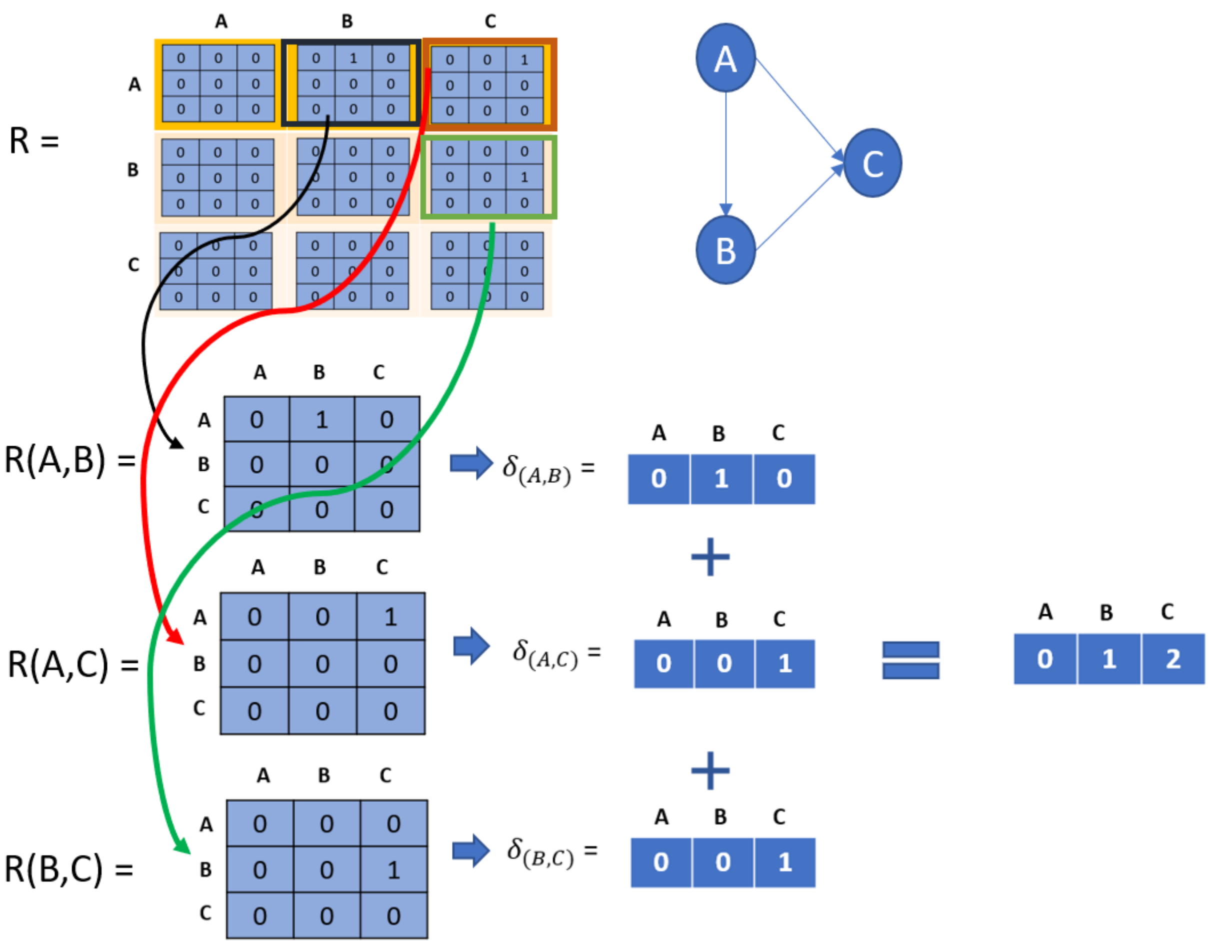

Figure 2.4: Computing the RBC by summing all the $delta_{st}$ vectors. The resulted RBC is the In-Degree of this graph.

# 3 Related Work

This section reviews other works in the field of complex network analysis that propose methods for learning centrality measures. The first part of this section will discuss methods that utilize traditional ML techniques for learning centrality measures. The second part will discuss methods that utilize more advanced Deep Learning techniques such as Graph Neural Networks (GNN) to learn centrality measures.

### 3.0.1 Learning centrality using traditional machine learning

In recent years, with the advancement in the machine and deep learning fields, researchers utilized many of these algorithms to approximate the centrality of graph's nodes.

In 2018, Grando et al. [12] presented methodology and associated ML-based techniques to effectively approximate node centrality measures. In general, they constructed a vector containing two features, where the first feature is the node's degree centrality, whereas the latter is the node's eigenvector centrality. They trained an NN model based on synthetic networks and the vectors obtained for predicting the node's centrality. Later, Grando et al. evaluated the performance of their model and compared it to other state-of-the-art ML algorithms, and applied the regression model obtained for real-world networks.

Mendonca et al. [13] improved the work presented by Grando et al. by proposing the NCA-GE model. The NCA-GE architecture utilizes Strucre2Vec and Graph Convolution Network (GCN) for generating a high dimensional feature vector for each node in a given graph. To these generated node embeddings, they added the degree centrality as an additional dimension. Finally, they utilized the embeddings obtained to approximate node centrality. They evaluated their approximations on synthetic and real-world networks by measuring the mean Kendall coefficient between their approximations and the actual centrality score. They presented better correlations in comparison to Grando et al.'s work.

Zhao et al. [14] focused on the problem of identifying influential nodes on complex graphs, which in many cases have a non-linear relationship between the functional importance of a node and its various features. In this scenario, measuring a single fixed centrality measure is not enough to detect the influential nodes in a given graph. Unlike traditional methods that evaluate a node's importance based on one or some global or local topologies, Zhao et al. proposed a framework that detects the importance of a node based on ML. The framework generated a feature vector of each node consisting of the values of 9 famous and classical centralities, such as degree, closeness, eigenvector, PageRank, etc., and the infection rate (which is an essential factor in the propagation scenarios). Then, they labeled each node based on the actual propagation ability obtained from their simulated propagation, which was based on SIR model [39]. They tested the model on different classic information scenarios and different sizes and types of networks. The results showed that, in general, ML methods outperformed the traditional centrality methods. Still, the accuracy of the ML methods was affected by both the infection rate and the number of labels. As opposed to Grando et al. [12], and Mendoncca et al. [13] works, this model considers the graph structure and not only fixed centrality measures.

It is important to mention that Grando et al. [12], Mendonca et al. [13], and Zhao et al. [14] considered other centrality measures in the pre-processing step. Thus, the model might have good results when evaluating correlated measures, but might not generalize well to approximate un-correlated measures or tasks that require learning new measures.

### 3.0.2 Learning centrality using deep graph neural networks

Maurya et al. [15] proposed a graph neural network (GNN) based model to approximate betweenness and closeness centralities. In GNN, a vector representation is generated for each node based on the aggregation of the adjacent nodes; therefore, these features utilize these characteristics to model paths and learn how many nodes are reachable to a specific node. The model receives the input graph and the adjacency matrix. Next, the graphs are pre-processed, and the adjacency matrix is modified such that nodes aggregate features over multiple hops along possible shortest paths in the given graph. The model learns a score function that maps the aggregated node's information to a score correlated with the centrality measure score. Two variants of the model GNN-Bet and GNN-Close for approximating betweenness and closeness were proposed. Each of them required a different pre-processing step and a different GNN structure. The experiments were conducted on a series of synthetic and real-world graphs, where both models were faster than other methods while providing higher ranking performance. In addition, the model training is inductive; it can be trained on one set of graphs and evaluated on another set of graphs with varying structures. While this approach requires different pre-processing and learning procedures for different centrality measures, LRC pre-processing and learning phases are identical for any learned centrality measure.

Fan et al. [40] proposed the DRBC model that focuses on the efficient identification of top k nodes with the highest SPBC in a graph. To identify the nodes with the highest SPBC, they used a graph neural network encoder-decoder ranking model. The model first encodes the nodes into embedding vectors capturing structural information for each node in the embedding space. Then, they use a decoder designed as a multi-layer perceptron (MLP) to compute the SPBC ranking score. Extensive experiments were conducted on synthetic networks and real-world networks from different domains. Fan et al. achieved comparable accuracy on both synthetic and real-world networks to state-of-the-art sampling-based baselines, but their model is far more efficient than sampling-based baselines for the running time. While this approach might be faster than LRC, it focuses solely on SPBC, while LRC is more generic and can be used to learn many centrality measures on graphs.

Avelar et al. [41] proposed a model that utilizes GNNs for multitasking learning and demonstrated the multitask learning capability of the model in the relevant relational problem of estimating network centrality measures, focusing primarily on producing rankings. The authors are concerned whether a neural network can approximate centrality measures solely from a network's structure. The suggested model trained an Multilayer Perceptron (MLP) which assigned with computing the probability that $v_i >_c v_j$ given their embeddings, where $>_c$ here denotes the total ordering imposed by the centrality measure c, that is, the node $v_i$ whose embedding is on the first d dimensions has higher c-centrality than $v_j$ whose embeddings is on the last d dimensions of the input. The proposed model obtains reasonable accuracy on a dataset of real-world instances with up to 4k nodes and achieves 89% accuracy on a test dataset of random instances with up to 128 nodes and is shown to generalize to larger problem sizes.

The paper has two main contributions: First, the experimental analyses of different GNN approach to ranking nodes and how well they scale in a multitasking environment, and the ability to predict centrality values solely from the network structure. Second, this model considers only the network structure to approximate centrality. But, the main focus of this work here is on multitasking learning and estimating centrality measure serves as an example.

# 4 Research Description

## 4.1 Research Hypothesis

This thesis deals with learning centrality measures from graph nodes' empirical measurements using the RBC algorithm. One of the inputs to the RBC algorithm is the routing policy. Creating a routing policy is not a trivial task because it requires a deep understanding of the centrality measure, complex network analysis capabilities, time, and effort. Thus, originally RBC was proposed as a generalization of betweenness measures only, meaning that one can build routing policies that will be compatible with betweenness measures computations.

**We hypothesize that RBC can be generalized to other centrality measures as well**. Our hypothesis is based on the assumption that we can utilize the power of neural networks to learn routing functions that produce routing policies that are compatible with other centrality measures rather than betweenness. The assumption is validated in the experiments by using our neural-network-based architecture LRC on multiple centrality measures, not only betweenness measures. This validation process was carried out in hopes that LRC will learn a routing function that produces a meaningful routing policy as input to the RBC algorithm for the centrality measure computation.

The assumption was tested and described in the following sections.

## 4.2 Methods

In this section, we describe how the Learned Routing Centrality (LRC) architecture was developed and explain the process of learning a routing function. First, we present the EigenvectorRBC algorithm - differentiable implementation of RBC. Then we discuss the direct optimization of the routing policy (that uses the EigenvectorRBC algorithm) to set intuition for LRC. Finally, we present the complete complete DL assembly that is used during the evaluation in section 5.

**Example 1 (Closeness as a special case of RBC)** *To showcase the feasibility of finding an appropriate routing policy to fit a given centrality measure, we provide a brief example of computing closeness using RBC with a tailor-made routing policy. Let $d(s,t)$ be the hop distance from s to t and $CC(v) = \sum_s d(s,v)$ be the closeness centrality. Let $R_{CC}(s,t,u,v)$ be a routing policy that inflates the flow originating from s by 1 for every hop in every direction away from s: $R_{CC}(s,t,u,v) = \frac{d(s,u)+1}{d(s,u)}$. Note that $R_{CC}(s,t,u,v)$ is not a probability in this case. $\delta_{st}(v)$ is equal to $d(s,t)$. Following Equation 2.1, $RBC = \sum_{s,t} \delta_{st} = \sum_{s,t} d(s,t) = n \cdot CC$.*

### 4.2.1 The Eigenvector RBC Algorithm

Combinatorial computation of RBC is based on a topological sort of the nodes according to the directed acyclic graph defined by $R(\otimes, t, u, v)$. Here and in the rest of this thesis, we use $\otimes$ to indicate any possible value of an argument. Topological sort is not differentiable and thus cannot be efficiently utilized within DL architectures to learn a routing function. In this section, we present a differentiable algebraic computation of RBC.

Consider a source node $s$ and a target node $t$. $R(s,t,\otimes,\otimes)$ (or $R(s,t)$ for short) defines a stochastic Markov transition matrix leading from $s$ to $t$. Following Dolev et al. [16], we require that $R(s,t,s,s) = 1$ for all $s, t \in V$ forming a self loop in every source node $s$. We also require $t$ to be the only sink in $R(s,t)$ without sink loops. We ensure that $R(\otimes,\otimes,u,v) = 0$ for all $u, v \notin E$ with a scalar multiplication of $R(s,t)$ and the adjacency matrix of the graph ($Adj(G)$). The principal eigenvector of $R(s,t)$ represents the steady-state probabilities of reaching each node on the way from $s$ to $t$ [42]. Thus, $\delta_{st}$ is the principal eigenvector of $R(s,t)$, and RBC vector can be computed using Equation 2.1.

Figure 4.1 shows an example of computing $\delta_{03}$. The algorithm for computing RBC using eigenvectors is summarized in Algorithm 1. It receives $R$, $T$, and $G$ as inputs and returns the RBC of all nodes. The complexity of the Eigenvector RBC algorithm is $O(n^4)$. All computations described in this thesis were implemented using PyTorch, allowing parallelization using GPUs.

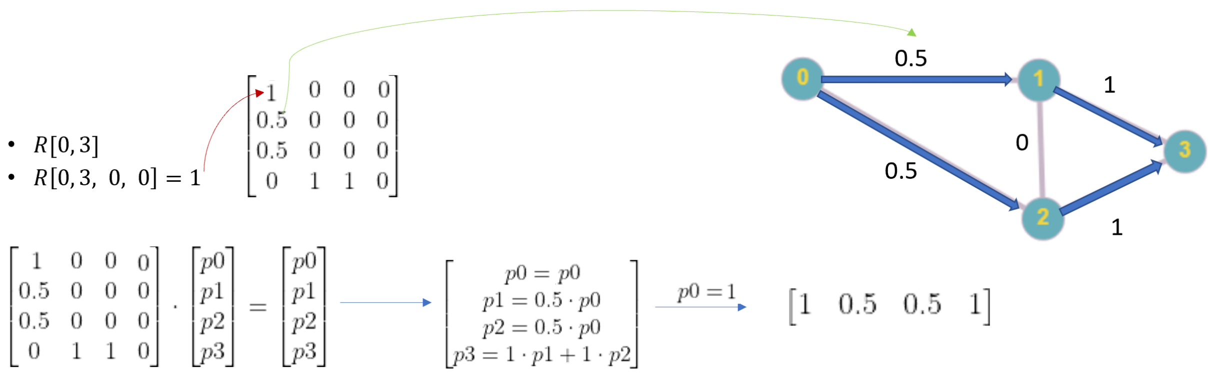

Figure 4.1: Example of computing $\delta_{03}$ as an eigenvector of $R(0,3)$

**Algorithm 1:** Eigenvector RBC

```
1  Input: ($R$: routing policy , $T$: traffic matrix, $G = (V, E)$: graph );
2  Output: ($RBC$: Centrality vector of the graph's nodes);
3  $\forall_{s,t \in V} R(s,t) = R(s,t) \cdot Adj(G)$;
4  $\forall_{s,t \in V} R(s,t,s,s) = 0$;
5  $\forall_v RBC(v) = 0$;
6  for $s, t \in V$ do
7      $R'(s,t) = R(s,t)$;
8      $R'(s,t,s,s) = 1$;
9      $\delta_{st} = Eigenvector(R'(s,t))$;
10     $RBC+ = \delta_{st} \cdot T(s,t)$;
11 end
12 return $RBC$
```

### 4.2.2 Direct Optimizationn of the Routing Policy

Next, we briefly discuss the direct optimization of the routing policy to set the intuition for the Learned Routing Centrality (LRC). The easiest way to fit the routing policy $R$ is by defining it as a 4D parameter matrix to be optimized the DL algorithms. In the rest of this thesis, the term routing policy refers to the 4D routing matrix or its slices.

In every epoch, we apply Algorithm 1 to compute RBC. Then, we use mean squared error (MSE) loss function to compare it to the target centrality (TC) measure. Algorithm 1 creates a computation tree connecting the RBC vector with MSE through differentiable operations. The loss is backpropagated through the computation tree to update the routing policy gradients and a DL optimization algorithm is applied. Fig. 4.2 presents the direct optimization of the routing policy.

The approach for direct optimization of the routing policy described here converges perfectly well. However, it does not generalize to multiple graphs because every index in the 4D matrix $R$ represents a particular vertex. While it is possible to use direct optimization to learn from a subset of nodes, we noticed during preliminary experiments that these approaches overfit the training data due to the large number ($mn^2$) of learned parameters compared to the maximal number of nodes ($n$) in the training set. Thus, we keep the above discussion mainly for didactic reasons and present a general solution to learning a routing function from node embeddings in the following subsection.

### 4.2.3 Learning the Routing Function from Node Embedding

Geometric graph embedding is a technique that positions nodes of a graph in a multi-dimensional Euclidean space such that nodes are connected if and only if they are at a distance at most $t$ for some threshold $t$ [43]. Not all graphs can be embedded in a low-dimensional space. In this thesis, we assume graphs that do. Simple heuristics, such as the compass routing [44] rely on node positions to efficiently navigate the graph. Inspired by this observation, we develop a DL-based routing function that can estimate the routing policy from node embeddings. In the rest of this thesis, the term routing function refers to a quarternary function that computes the probability that $u$ will forward to $v$ a packet sent from $s$ to $t$ from the embeddings of $s, t, u, v$. To simplify the notation, we denote routing policies and routing functions by $R$ and use the complete terms in ambiguous contexts.

Instead of explicitly maintaining the 4D routing policy, we suggest using a fully connected deep neural network with a smaller number of parameters to estimate the routing policy from node embeddings. Fig. 4.3 presents the LRC architecture and Algorithm 2 summarizes the computation steps. Similar to Section 4.2.2, we use Algorithm 1 to compute Eigenvector RBC from a routing policy. The main differences between LRC and the architecture described in Section 4.2.2 are the node embedding and the routing function applied before Eigenvector RBC.

LRC instances are collections of 4-tuples where each entry is a $k$-dimensional node embedding. The routing function module can be considered a convolution applied on all the 4-tuples to produce a single output – the routing policy. We use a sparse matrix to store the routing policy since vast majority of its entries correspond to non-edges and are nullified.

LRC can be trained on many graphs of arbitrary sizes as long as the node positions follow the same geometric rules. The actual positions of the nodes do not matter. Thanks to the variety of graphs in the training set, and their scattered positions in the embedding space, the routing function learns to infer the routing policy entries from the relationships between the node embeddings, e.g., learns distances of sorts.

The loss function used to train the LRC depends on the task at hand. Since, in many cases, the correlation between the computed and the fundamental importance of nodes is more important than the actual values, we suggest using one minus Pearson correlation as the LRC loss function. Unfortunately, rank correlations such as Spearman's cannot be incorporated into LRC architecture due to lack of differentiable implementations.

The training loop may be halted after a predefined number of epochs or stopped earlier when the correlation between LRC and the target centrality measure is sufficiently high. It is important to include several different graphs in a batch. Applying the optimization step after every graph requires careful hyper-parametr optimization of the optimizer to ensure convergence.

**Algorithm 2:** LRC Training

```
1  Input: ($Gs$: Training graphs, $TCs$: target centrality vectors, $Em$:
     embedding function);
2  $R \leftarrow$ randomly initialize the routing function;
3  $\forall_{g \in Gs}$ compute $Em(g)$;
4  $loss = \infty$;
5  while $loss \geq \epsilon$ do
6      loss = 0;
7      for $g = (V_g, E_g) \in Gs$ do
           /* Select the embedding 4-tuples                    */
8          $STUV = \{(Em(s), Em(t), Em(v), Em(u)) : s, t, u, v \in V_g\}$;
           /* Compute the routing policy                       */
9          $RP = R(SUTV)$;
           /* Compute the routing centrality                   */
10         $LRC = EigenvectorRBC(RP, T, g)$;
           /* Aggregate errors                                 */
11         $loss+ = 1 - Pearson(LRC, TCs(g))$;
12     end
       /* Optimization step                                    */
13     backpropagate $loss$;
14     optimize $R$;
15 end
```

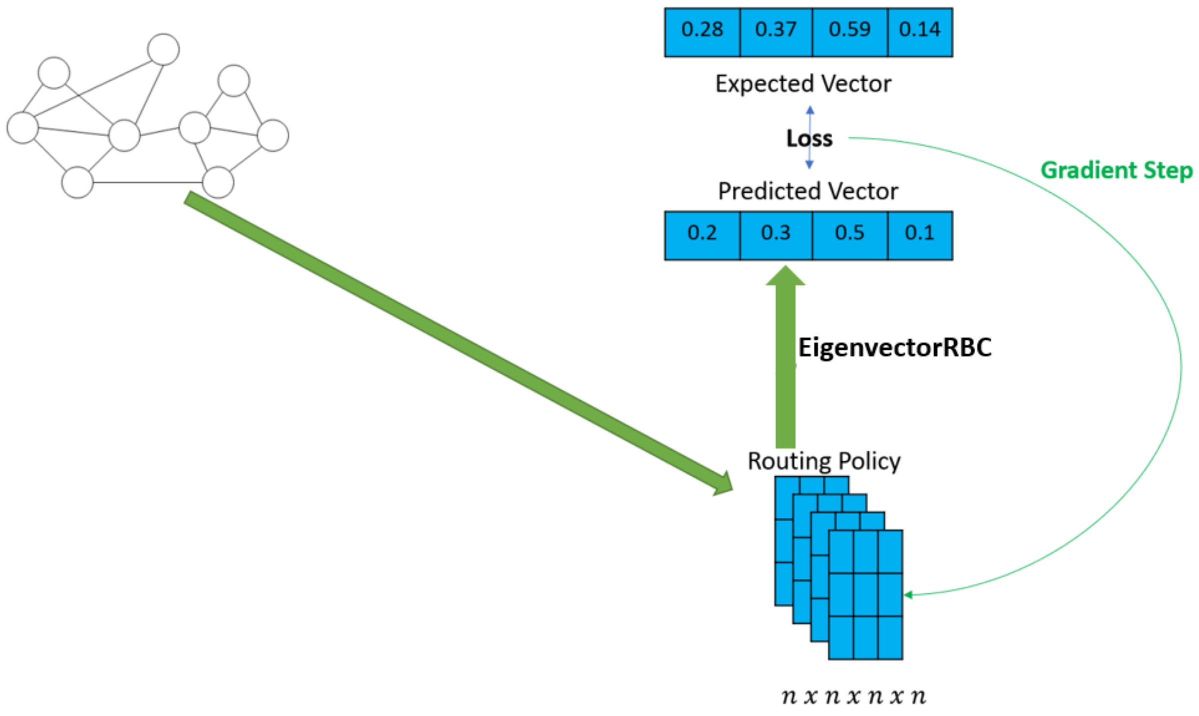

Figure 4.2: The architecture for Direct optimization of the routing policy.

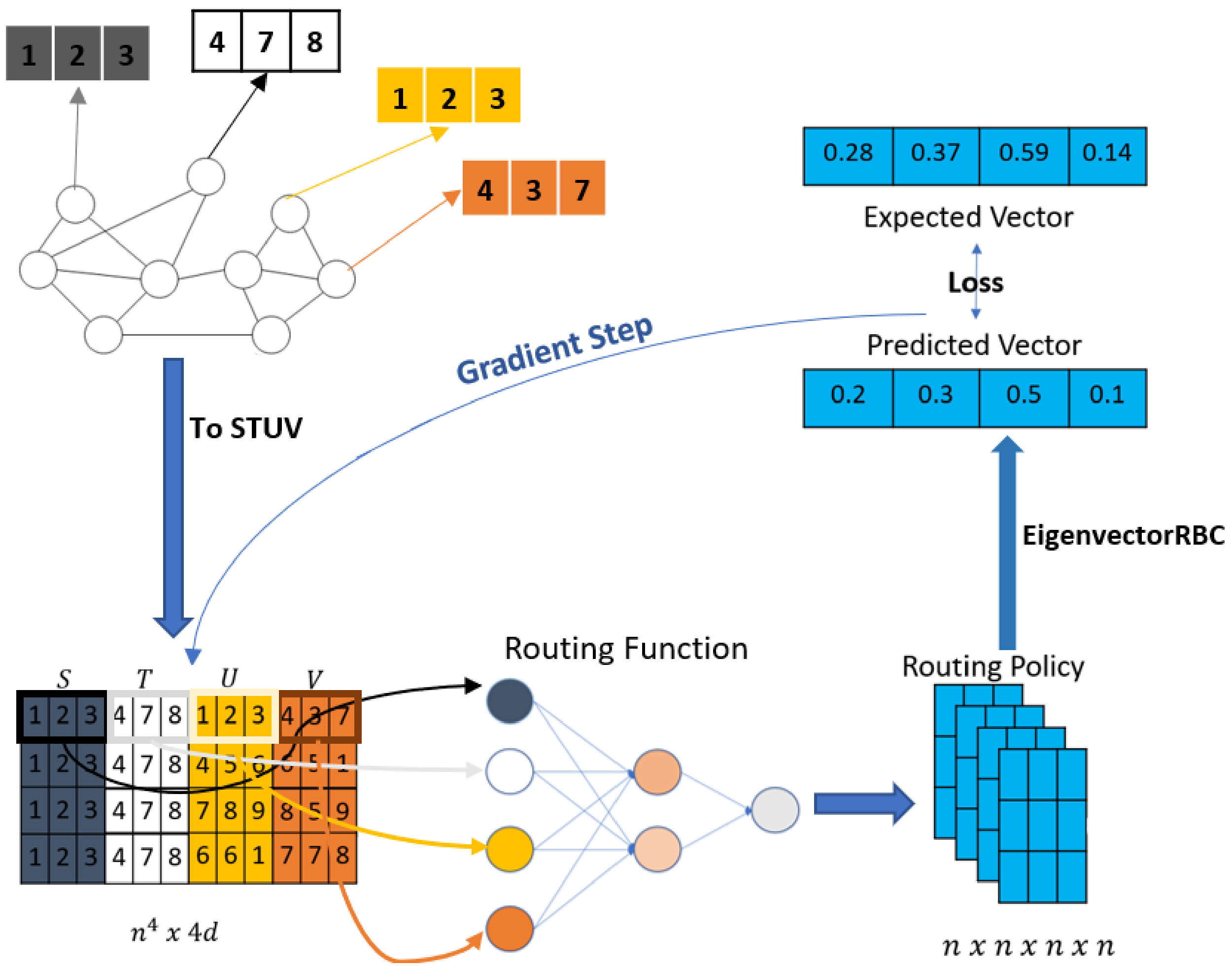

Figure 4.3: The architecture of LRC. learning the routing function from node embedding using a collection of 4-tuples.

# 5 Evaluation

## 5.1 Experiment with arbitrary graphs

The goal of LRC is to learn a tailored routing function for a given task. Using the learned routing function, one can build the routing policy of a given graph. The routing policy then is used by the EigineVectorRBC (Alg 1) to produce the node's centrality vector. One of the most important attributes of the routing function is generality: using the learned routing function, one can build a near-optimal routing policy for multiple types of graphs. To achieve generality, the dataset should include different graph types when training a model with LRC. Hence, in the first experiment in this thesis, the models were trained with the LRC architecture on arbitrary graphs.

### 5.1.1 The embedding method

The first step in LRC training, as depicted in algorithm 2 is to generate node embedding. In this thesis, we investigated multiple node embedding methods and decided to use the GLEE [20] node embedding method. The main reason for using GLEE relates to the fact that GLEE uses the Laplacian matrix to find an embedding with geometric properties instead of spectral ones. LRC uses geometric properties while learning (computation of eigenvalues and eigenvectors). Moreover, geometric properties are more suitable for learning routing between nodes, as LRC learns, than spectral properties. Fig 5.1 presents an example for GLEE node embedding on a graph with two massive clusters and a few nodes between these clusters. We generated 2D embedding for this graph with three different seeds. As can be seen, GLEE successfully separated between the two massive clusters and also the nodes between these clusters.

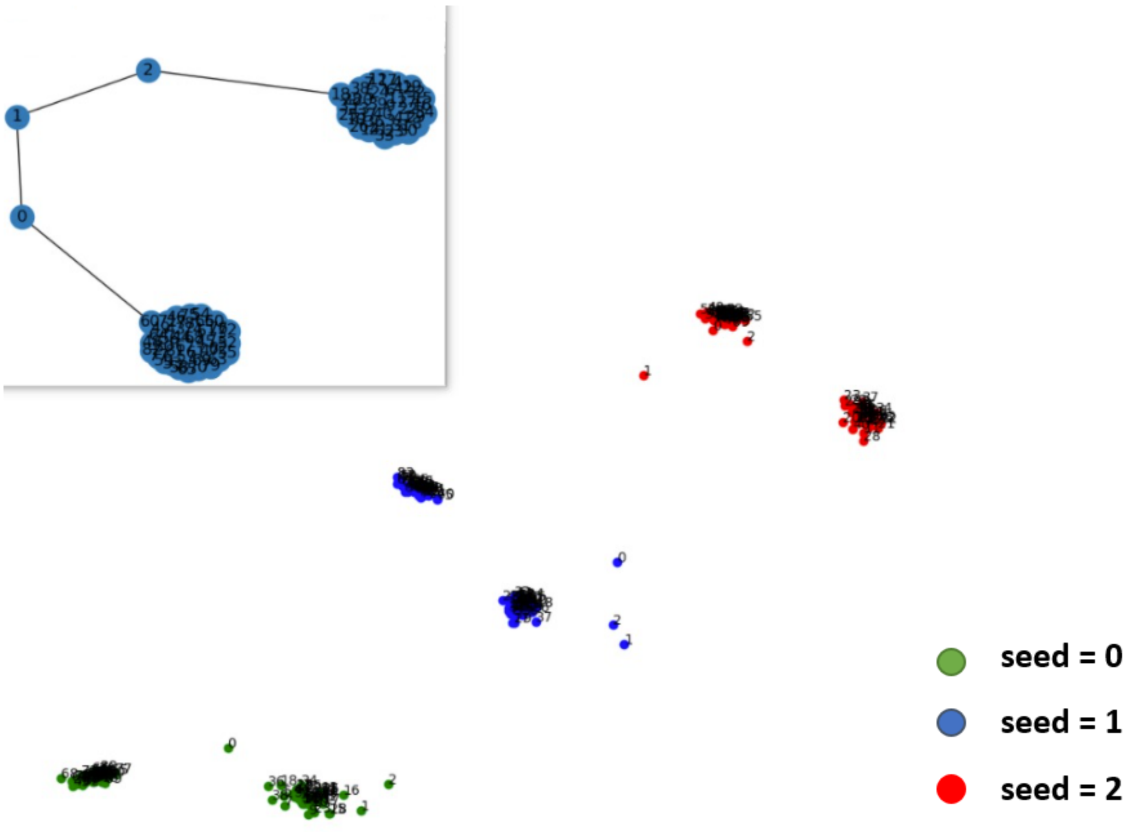

Figure 5.1: GLEE 2D node embedding example

### 5.1.2 Experimental Setup

To explore LRC performance on arbitrary graphs, we generated synthetic graphs with the NetworkX graph generator package [45]. The type of graph generated included $G_{n,p}$ random graph (also called Erdős-Rényi graph), $G_{n,m}$ random graph, binominal graph, Newman-Watts, Barabási–Albert graph, random kernel graph, Barabási–Albert model graph. The training set included 150 graphs, and the test set 30 graphs where the ratio between the number of graphs from each type was similar in the train and the test set. The node embedding technique was GLEE. Finally, to evaluate the proposed LRC technique, we trained a model using LRC for learning the *Load Centrality* measure.

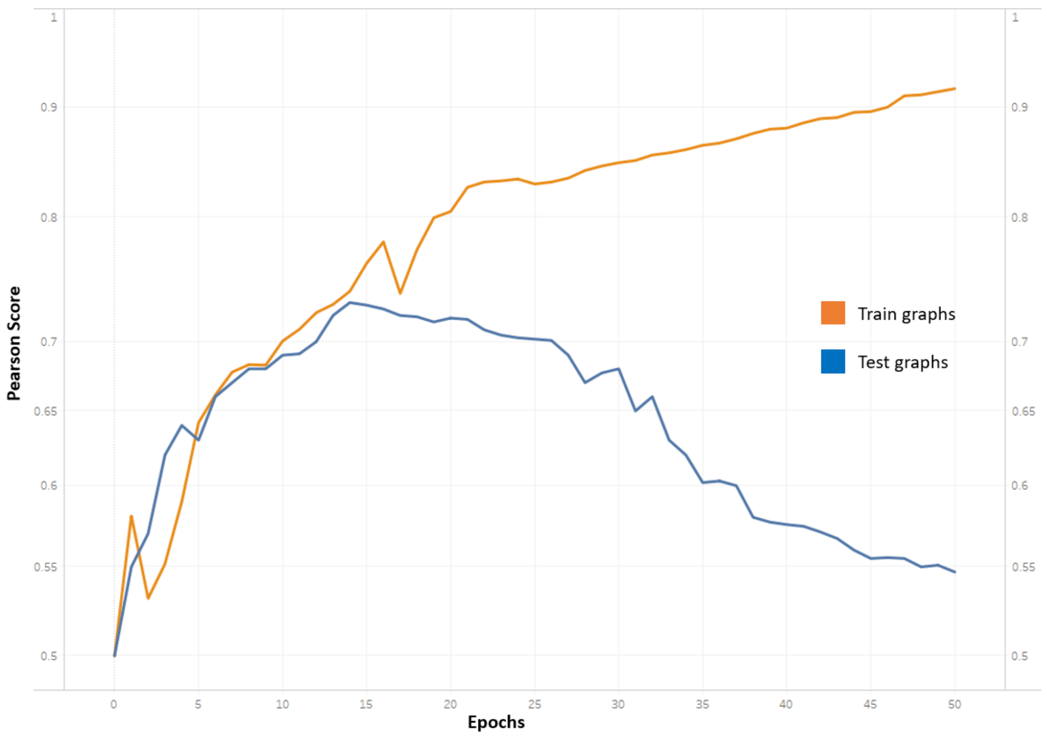

Figure 5.2: Pearson correlation score during 50 epochs on arbitrary graphs

### 5.1.3 Results

During 50 training epochs, we tested the correlation between LRC and the Load centrality measure on the test graphs. Unfortunately, this experiment resulted in overfitting. Figure 5.2 shows that pearson correlation gradually improves on both train and test sets until epoch 15, but after that point, the correlation score keeps improving on the training set while it decreases on the test set.

## 5.2 Experiments with Random Geometric Graphs

After an unsuccessful attempt to learn centrality measures on arbitrary graphs with the LRC architecture and the GLEE node embedding method, we shifted our focus from arbitrary graphs to random geometric graphs (RGG) for the following reasons:

1. Geometric graphs have a natural embedding - the node position in the topological space. Thus, we can utilize this embedding instead of using an embedding function.

2. Geometric graphs optimized for finding the shortest path in a network [46]. As Doelv et al [16] stated RBC is a generalization of betweenness measures, and the routing policy for computing betweenness measure with the RBC algorithm is based on the shortest paths in a graph. We hypothesize that LRC can utilize the structure of geometric graphs to learn routing functions that produce routing policies based on the shortest paths in a graph.

### 5.2.1 Experimental Setup

The experiments were conducted on random geometric graphs (RGGs) with dozens of nodes. The nodes were distributed over a square region of size $1 \times 1$ and a radius length of 0.4. The training set included 200 graphs while the test set included 32 graphs. To evaluate the proposed LRC technique, we trained five models using LRC for learning five centrality measures: Load Centrality (Load), Shortest Path Betweenness Centrality (SPBC), closeness, degree, and eigenvector centralities on the various graphs. Next, we utilized the trained models for predicting the given centrality measure score for each node in each graph in the test set. Later, we measured the correlation between the LRC scores obtained and the actual centrality scores using the SciPy statistics package [47]. The correlations measured were Kendall, Pearson, and Spearman. To compare the performance of LRC, we measured the centrality scores using each of the centrality measures, as well as the state-of-the-art NCA-GE approach suggested by Mendonca et al. [13].

To compare LRC with NCA-GE, we used node embeddings identical to ours, node position in the topological space. Mendoncca et al. [13] mentioned that the embedding technique is available for user selection. In addition, we used the same NN structure (except for the input layer) and the same training hyperparameters. Also, in the preprocessing phase, the node degree was concatenated with the node embedding vector as described.

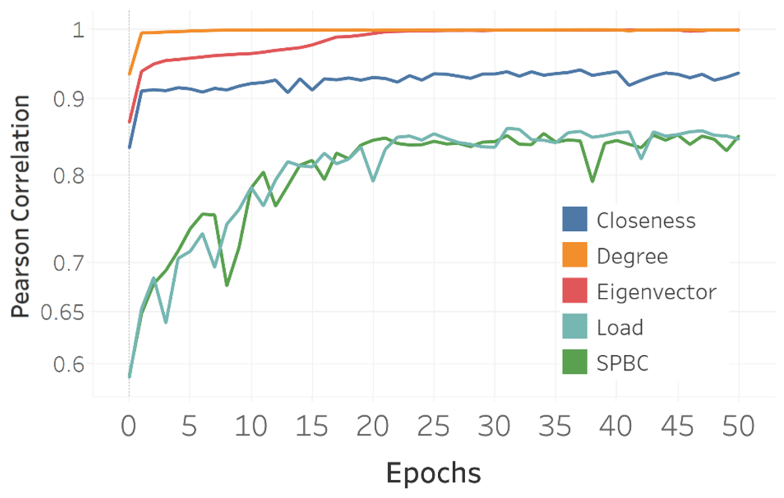

Figure 5.3: Pearson correlation as a function of epochs on the test graphs on various centrality measures.

### 5.2.2 Results

#### Learning various centrality measures with LRC

**It learns!** During 50 training epochs, we tested the correlation between incumbent LRC and the target centrality measures on the test graphs. Figure 5.3 shows that Pearson correlation gradually improves and stabilizes after epoch 20. It demonstrates the ability of the model to learn the routing function and the convergence of the learning process.

LRC obtained a Pearson correlation of 0.85 with SPBC and Load, 0.936 with closeness, and the highest Pearson correlation of 0.999 with eigenvector centrality and degree. Degree centrality was found to be the easiest to learn by LRC as seen from its fastest convergence.

#### Comparing LRC to baseline

Table 5.1 presents the Kendall, Pearson, and Spearman correlation scores between the target centrality measures and their alternatives on the test graphs. These three correlation scores were chosen because they were used in previous studies. Two centrality learning techniques are evaluated: the state-of-the-art NCA-GE [13] as a baseline and our proposed LRC. Correlations with other centrality measures are also provided as baselines

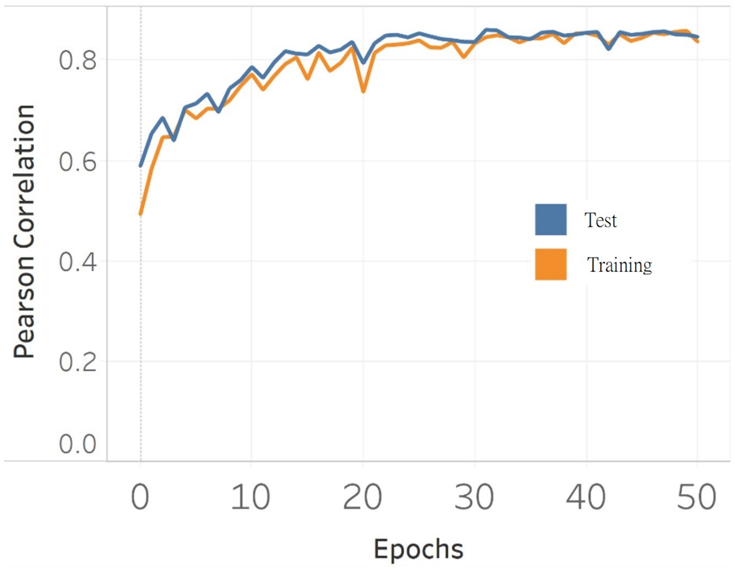

Figure 5.4: The Pearson correlation score when learning Load centrality, on the train and test graphs.

because all centrality measures naturally correlate with each other.

## 5.3 Discussion

As can be seen from Figure 5.2 LRC fails to learn on the dataset of arbitrary graphs, and the learning process results in over-fitting. But, when we shift our focus to RGG we can see LRC successfully learns, Fig 5.3 shows that LRC converges well on betweenness, closeness, degree, and eigenvector centralities. We believe that eigenvector centrality is found to be learned quickly since Eigenvector RBC relies on Eigenvectors for computing RBC. Surprisingly, the routing functions of SPBC and Load centrality were found the most difficult to learn albeit RBC is a variant of betweenness centrality and was originally designed to compute these measures. The disadvantageous performance of LRC on betweenness can be attributed to over-fitting. We reject this explanation by comparing LRC performance on training and on test graphs (see Fig. 5.4). Consequently, we attribute such behavior to the mismatch between Euclidean distances, supposedly learned by the routing function when optimized for betweenness, and hop distances between the nodes.

| Correlation | | SPBC | Closness | Degree | Eigenvector | NCA-GE | LRC |
|---|---|---|---|---|---|---|---|
| Kendall | SPBC | | **0.71** | 0.65 | 0.48 | 0.496 | **0.71** |
| | Closeness | 0.71 | | **0.83** | 0.73 | 0.79 | 0.82 |
| | Degree | 0.65 | 0.83 | | 0.83 | | **0.91** |
| | Eignvector | 0.48 | 0.73 | 0.83 | | 0.73 | **0.96** |
| Pearson | SPBC | | 0.76 | 0.66 | 0.48 | 0.58 | **0.85** |
| | Closeness | 0.76 | | 0.9 | 0.83 | 0.91 | **0.93** |
| | Degree | 0.66 | 0.9 | | 0.91 | | **0.99** |
| | Eignvector | 0.48 | 0.83 | 0.91 | | 0.89 | **0.99** |
| Spearman | SPBC | | 0.82 | 0.74 | 0.58 | 0.61 | **0.83** |
| | Closeness | 0.82 | | 0.89 | 0.81 | 0.88 | **0.9** |
| | Degree | 0.74 | 0.89 | | 0.9 | | **0.96** |
| | Eignvector | 0.58 | 0.81 | 0.9 | | 0.84 | **0.988** |

Table 5.1: Comparing LRC to baseline, NCA-GE was not compared with degree because it uses degree as a feature. Maximal correlation scores are highlighted with bold.

As for comparing LRC to the state-of-the-art ML technique for learning centrality measures NCA-GE, Table 5.1 Shows that LRC outperforms NCA-GE on multiple centralities (on the RGG dataset). It is important to mention that although the LRC is optimized to maximize the Pearson correlation, as a side-effect, the Kendall and Spearman correlations are also high.

# 6 Conclusion

In this thesis, we have presented a differentiable algebraic computation of Routing Betweenness Centrality (RBC). We discussed the ability of RBC to represent arbitrary, not necessarily betweenness-like centrality measures and proposed a DL architecture to compute the Learned Routing Centrality (LRC). First, we trained the LRC architecture on arbitrary graphs with the GLEE node embedding method. Unfortunately, the experiments with arbitrary graphs resulted in overfitting. Then, we shifted the focus of our efforts to geometric graphs, utilizing the fact that they optimized for finding the shortest path (euclidean distance) in a network. And indeed, the experiments performed with geometric graphs showed that LRC converges well, reaching high correlations with betweenness, closeness, degree, and eigenvector centralities. Moreover, we compared LRC with the state-of-the-art ML technique for learning centrality measures NCA-GE, and as can be seen from table 5.1 LRC outperforms NCA-GE on multiple centrality measures.

Here, the high time complexity of EigenvectorRBC limits the size of the graphs on which centrality learning could be demonstrated. We partially mitigate this limitation by efficiently parallelizing the computation thanks to PyTorch's GPU support. Further research is required to improve the complexity of differentiable centrality computation to utilize the full power of deep learning on graphs.

This thesis focuses on geometric graphs where routes can be efficiently estimated from node positions. A follow-up investigation is required to learn a node embedding function alongside the routing function to apply the proposed approach on arbitrary graphs. We see great potential in using geometric embedding algorithms to produce node positions suitable for applying LRC or arbitrary graphs. Later on, explainable artificial intelligence algorithms can be used to visualize the learned routing functions and understand the structural properties that facilitate target variables, such as social influence. We believe that the presented architecture and insights provide an important step toward a generic framework for learning arbitrary centrality measures.

# Bibliography

[1] Donghyeon Yu, MinSoo Kim, Guanghua Xiao, and Tae Hyun Hwang. Review of biological network data and its applications. *Genomics & informatics*, 11(4):200, 2013.

[2] Gerta Rücker. Network meta-analysis, electrical networks and graph theory. *Research synthesis methods*, 3(4):312–324, 2012.

[3] Olaf Sporns. Graph theory methods: applications in brain networks. *Dialogues in clinical neuroscience*, 20(2):111, 2018.

[4] Meytal Tubi, Rami Puzis, and Yuval Elovici. Deployment of dnids in social networks. In *2007 IEEE ISI*, pages 59–65. IEEE, 2007.

[5] Rami Puzis, Dana Yagil, Yuval Elovici, and Dan Braha. Collaborative attack on internet users' anonymity. *Internet Research*, 2009.

[6] Kundan Kandhway and Joy Kuri. Using node centrality and optimal control to maximize information diffusion in social networks. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 47(7):1099–1110, 2017.

[7] Francisco Aparecido Rodrigues. *Network Centrality: An Introduction*, pages 177–196. Springer International Publishing, Cham, 2019.

[8] Xingqin Qi, Eddie Fuller, Qin Wu, Yezhou Wu, and Cun-Quan Zhang. Laplacian centrality: A new centrality measure for weighted networks. *Information Sciences*, 194:240–253, 2012.

[9] Alireza Abbasi and Liaquat Hossain. *Hybrid Centrality Measures for Binary and Weighted Networks*, pages 1–7. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[10] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[11] William L Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on NIPS*, pages 1025–1035, 2017.

[12] Luis C. Lamb Felipe Grando, Lisandro Z. Granville. Machine learning in network centrality measures: Tutorial and outlook. *ACM Computing Surveys, Vol. 51, No. 5, Article 102*, 2018.

[13] Matheus RF Mendonça, André Barreto, and Artur Ziviani. Approximating network centrality measures using node embedding and machine learning. *ACMVol 57*, 2020.

[14] Gouheng Zhao, Peng Jia, Cheng Huang, Anmin Zhou, and Yong Fang. A machine learning based framework for identifying influential nodes in complex networks. *IEEE Access 10.1109/ACCESS.2020.2984286*, 2020.

[15] Sunil Kumar Maurya, Xin Liu, and Tsuyoshi Murata. Graph neural networks for fast node ranking approximation. *TKDD 2021*, 15(5):1–32.

[16] Rami Puzis Shlomi Dolev, Yuval Elovici. Routing betweenness centrality. In *Routing Betweenness Centrality*, page 27. IEEE, 2010.

[17] J. M. ANTHONISSE. The rush in a directed graph. *Tech. rep. BN 9/71, Stichting Mathematisch Centrum, Amsterdam, The Netherlands.*, 1971.

[18] Mark EJ Newman. Scientific collaboration networks. ii. shortest paths, weighted networks, and centrality. *Physical review E*, 64(1):016132, 2001.

[19] Liav Bachar, Aviad Elyashar, and Rami Puzis. Learning centrality by learning to route. In *International Conference on Complex Networks and Their Applications*, pages 247–259. Springer, 2021.

[20] Leo Torres, Kevin S Chan, and Tina Eliassi-Rad. Glee: geometric laplacian eigenmap embedding. *Journal of Complex Networks*, 8(2):cnaa007, 2020.

[21] Bavelas A. A mathematical model for group structure. *Human Organization 7, 16–30*, 1948.

[22] Bavelas A. Communication patterns in task orientated groups. *Journal of the Acoustical Society of America 22, 271–288*, 1950.

[23] Shimbel A. Structural parameters of communication networks. *Bulletin of Mathematical Biophysics, 15, 501- 507.*, 1953.

[24] C. Mitchell, R. Agrawal, and J. Parker. The effectiveness of edge centrality measures for anomaly detection. pages 5022–5027, 2019.

[25] A. Srinivas and R. L. Velusamy. Identification of influential nodes from social networks based on enhanced degree centrality measure. pages 1179–1184, 2015.

[26] Rami Puzis, Yaniv Altshuler, Yuval Elovici, Shlomo Bekhor, Yoram Shiftan, and Alex (Sandy) Pentland. Augmented betweenness centrality for environmentally aware traffic monitoring in transportation networks. *Proceedings of the ACM Conference on Online Social Networks*, 2013.

[27] M. E. J. Newman. The mathematics of networks. *Center for the Study of Complex Systems, University of Michigan, Ann Arbor, MI 48109–1040*, 2006.

[28] L. C. FREEMAN. A set of measures of centrality based on betweenness. *Sociometry 40, 1, 35–41.*, 1977.

[29] GOH, KAHNG, and KIM D. Universal behavior of load distribution in scale-free networks. *Phys. Rev. Lett. 87, 27 (Dec.), 278701.*, 2001.

[30] M. Kendall. Rank correlation methods., 1948.

[31] Marie-Therese Puth, M. Neuhäuser, and G. Ruxton. Effective use of spearman's and kendall's correlation coefficients for association between two measured traits. *Animal Behaviour*, 102:77–84, 2015.

[32] Yuan Fang, Wenqing Lin, Vincent W Zheng, Min Wu, Kevin Chen-Chuan Chang, and Xiao-Li Li. Semantic proximity search on graphs with metagraph-based learning. In *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*, pages 277–288. IEEE, 2016.

[33] Zemin Liu, Vincent W Zheng, Zhou Zhao, Fanwei Zhu, Kevin Chen-Chuan Chang, Minghui Wu, and Jing Ying. Semantic proximity search on heterogeneous graph by proximity embedding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

[34] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710, 2014.

[35] Karel Devriendt and Piet Van Mieghem. The simplex geometry of graphs. *Journal of Complex Networks*, 7(4):469–490, 2019.

[36] Miroslav Fiedler. *Matrices and graphs in geometry*. Number 139. Cambridge University Press, 2011.

[37] Chapter 5 - malware-propagative markov random fields. In Vasileios Karyotis and M.H.R. Khouzani, editors, *Malware Diffusion Models for Wireless Complex Networks*, pages 107–138. Morgan Kaufmann, Boston, 2016.

[38] Ulrik Brandes. A faster algorithm for betweenness centrality. *Journal of mathematical sociology*, 25(2):163–177, 2001.

[39] W. O. Kermack and A. G. McKendrick. A contribution to the mathematical theory of epidemics. *arXiv:1809.07695v4*, 2019.

[40] Changjun Fan, Li Zeng, Yuhui Ding, Muhao Chen, Yizhou Sun, and Zhong Liu. Learning to identify high betweenness centrality nodes from scratch: A novel graph neural network approach. In *CIKM 2019*, pages 559–568.

[41] Pedro H.C. Avelar, Henrique Lemos, Marcelo O.R. Prates, and Luis C. Lamb. Multitask learning on graph neural networks: Learning multiple graph centrality measures with a unified network. *Proc. Roy. Soc. London A, Containing Papers Math. Phys. Character, vol. 115, no. 772, pp. 700–721, 1927*, 1927.

[42] Taher Haveliwala, Sepandar Kamvar, Dan Klein, Chris Manning, and Gene Golub. Computing pagerank using power extrapolation. Technical report, Stanford, 2003.

[43] Jan Reiterman, Vojtech Rödl, and E Šinajová. Geometrical embeddings of graphs. *Discrete Mathematics*, 74(3):291–319, 1989.

[44] Evangelos Kranakis, Harvinder Singh, and Jorge Urrutia. Compass routing on geometric networks. In *CCCG 1999*. Citeseer.

[45] NetworkX. Networkx graph generator.

[46] Vijay Sundararajan. *Gate Sizing*, pages 811–814. Springer New York, New York, NY, 2016.

[47] SciPy community. Statistical functions(scipy.stats).

# תקציר

בתורת הגרפים, מדדי מרכזיות מאפשרים לדרג את חשיבות הקודקדים בגרף בהינתן משימה מסויימת. דוגמאות למשימות כאלו הן שליטה וניטור תעבורה ברשתות תקשורת, מציאת של משתמשים בעלי השפעה גבוהה ברשתות חברתיות, וזיהוי בוטים באינטרנט. כדי שדירוג החישבות של הקודקודים יהיה משמעותי, חשוב לבחור במדד מרכזיות מתאים למשימה. למרות שחוקרים המציאו מגוון מדדי מרכזיות לאורך השנים, מדי פעם צצה משימה חדשה שאין עבורה מדד מרכזיות מספיק טוב, ובמקרה כזה נדרש להמציא מדד מרכזיות חדש המותאם אישית למשימה.

יצירת מדד־מרכזיות עבור משימה ספציפית דורש מומחיות בתחום, ידע בניתוח רשתות מורכבות, זמן ומאמץ. לכן, היכולות ללמוד באופן אוטומטי מדד־מרכזיות עבור משימה ספיצפית הוא כיוון מחקר מאוד חשוב.

בתזה הזו, אנו מציעים ארכיטקטורה גנרית המבוססת על רשתות נוירונים שמטרתה ללמוד מדדי־מרכזיות, כאשר אנו מסתמכים על התובנה שניתן לחשב מדדי מרכזיות שרירותיים ע״י שימוש באלגוריתם 'ניתוב בין מרכזיות'. כדי לשלב את אלגוריתם 'ניתוב בין מרכזיות' עם הארכיטקטורה שלנו המבוססת על רשתות נוירונים, הצענו גרסא חדשה לאלגוריתם כך שהגרסא הזו גזירה. בשלב הניסויים, יצרנו בעזרת הארכיטקטורה מספר מודלים עבור מגוון מדדי־מרכזיות. התוצאות מראות שהמודלים שנוצרו בעזרת הארכיטקטורה שלנו, הצליחו להשיג קורלוציה גבוהה יותר עבור דירוג המרכזיות של הקדוקדים בהשוואה לשיטות הכי חדישות בתחום.

אוניברסיטת בן־גוריון בנגב
הפקולטה למדעי ההנדסה
המחלקה להנדסת מערכות תוכנה ומידע

# למידת מרכזיות ע”י למידת ניתוב

חיבור זה מהווה חלק מהדרישות לקבלת תואר ”מגיסטר“ בהנדסה

מאת: ליאב בכר

מנחים: ד”ר רמי פוזיס (PUZIS@BGU.AC.LI)
מר. אביעד אליאשר (AVIADEL2@SCE.AC.IL)

חתימת המחבר: [illegible] תאריך: 17/04/2022

אישור המנחה: ______________ תאריך: ______________

אישור יו”ר ועדת תואר שני מחלקתית: __________ תאריך: __________

אפריל 2202

אוניברסיטת בן־גוריון בנגב
הפקולטה למדעי ההנדסה
המחלקה להנדסת מערכות תוכנה ומידע

# למידת מרכזיות ע"י למידת ניתוב

ליאב בכר

חיבור זה מהווה חלק מהדרישות לקבלת תואר "מגיסטר" בהנדסה

מאת: ליאב בכר (BACHARLIAV@GMAIL.COM)

אפריל 2022