Machine Learning for Graph Algorithms and Representations

Allison Gunby-Mann
Dartmouth College, allison.mann.th@dartmouth.edu

Machine Learning for Graph Algorithms and Representations

A Thesis
Submitted to the Faculty
in partial fulfillment of the requirements for the
degree of

Doctor of Philosophy

in

Engineering Sciences

by Allison Mann

Thayer School of Engineering
Guarini School of Graduate and Advanced Studies
Dartmouth College
Hanover, New Hampshire

MAY 2024

Examining Committee:

Chairman ____________________
Peter Chin

Member ____________________
Vikrant Vaze

Member ____________________
Colin Meyer

Member ____________________
Wayne Snyder

____
F. Jon Kull, Ph.D.
Dean of Guarini School of Graduate and Advanced Studies

ABSTRACT

This thesis explores a variety of common graph theoretic problems from a machine learning perspective. The topics covered include fundamental network problems such as distance approximation, distance sensitivity, community detection, cross-network alignment, and graph embedding dimension reduction. These projects are unified by the theme of machine learning on graphs, graph embeddings, and representations of graphs.

Acknowledgements

I am deeply indebted to my husband Benjamin for his patience, understanding, and constant encouragement. His belief in me has been a source of strength and motivation.

I would like to express my heartfelt gratitude to my family, especially my mom, dad, brother, and my grandparents for their unwavering support and encouragement throughout this journey. My friends Chelsea, Wendy, Polina, and Scott also were a bastion for me during the entirety of my PhD.

I also want to extend my sincere appreciation to my wonderful advisor, Peter Chin, for his tireless and unwavering support. Words cannot express everything he has done for me in the past few years, from personally to professionally.

I would like to thank my committee members Vikrant Vaze, Colin Meyer, and Wayne Snyder for their guidance, feedback, and invaluable insights, which have been instrumental in shaping this work.

I am also grateful to my dedicated labmates both at Dartmouth College and at Boston University for their camaraderie, collaboration, and intellectual exchange. Their enthusiasm and companionship have enriched my research experience.

Lastly, I would like to acknowledge the contributions of all those who have collaborated with me in this endeavor. Especially my co-authors Ike Abioye, Davin Jeong, Sarel Cohen, Felicia Schenkelberg, and many others. Your encouragement and assistance have been integral to the completion of this thesis.

Contents

List of Tables

List of Figures

Chapter 1

Introduction and Purpose

Graph theory is an extremely versatile field in mathematics and computer science, often studied due to its wide range of applicability across a spectrum of domains. From modeling road networks to analyzing social connections and cyber infrastructures, graphs serve as powerful representations of complex systems. Traditionally, graph theory has tackled fundamental challenges over graphs such as distance computation, node classification, and clustering. Approaches as early as the 18th century with the Seven Brides of Konigsberg modeled the streets of a city as a graph and Euler concluded that it was impossible to cross each bridge in the city exactly once using elementary graph theory [3]. However, it wasn't until the 20th century when significant developments began. For example, in 1956 Dijkstra's algorithm was developed by Dutch researcher Edsger Dijkstra [4]. It is one of the fundamental algorithms in graph theory, solving the problem of shortest path distance computation from a designated start vertex to all other vertices in the graph. Researchers worked on algorithms to find shortest paths, spanning trees, network flows, matching problems, graph traversal, and optimization on graphs. Classic algorithms like Dijkstra's Algorithm, Prim's Algorithm, Kruskal's Algorithm, and Ford-Fulkerson Algorithm were developed to address these problems [3].

However, with the popularity and technical growth of machine learning, there has been a notable shift towards leveraging machine learning techniques to enhance our understanding and address not only classic graph problems but also new and more complex problems that arose from the digital age and increased data size and accessibility.

Traditional machine learning methods typically operate over vectors rather than graph structures. To bridge this gap, we employ graph representations in the form of vector space embeddings. In general, graph embeddings are used to generate low-dimensional representations of nodes while retaining properties of the original graph. Methods include matrix factorization, random walks, and generative models [5]. Whether optimizing embeddings for specific tasks or employing general embedding techniques like Node2Vec, these graph embeddings may then be integrated with existing methods to address tasks such as node classification, node clustering, and link prediction [6, 7].

1.1 Graph Applications

There are a few main categories of graph problems that are widely studied, most of which will be touched upon in this thesis:

1. **Shortest Path Computation** : The shortest path problem is frequently encountered in the real-world. In road networks, users want to know how long it will take to get from one place to another [8]. In biological networks, consisting of genes and their products, the shortest paths are used to find clusters and identify core pathways [9]. In social networks, the number of connections between users can be used for friend recommendation [10]. In web search, relevant web pages can be ranked by their distances from queried terms [11].

2. **Node/Edge Classification**: Node classification is a common application in graph analysis where the task involves assigning labels to nodes in a graph based on certain features or characteristics. In social networks, this could mean classifying spam vs non-spam accounts [12]. For cybersecurity, there is ample interest in detecting

malicious hosts, users, or activity which can be represented as nodes or edges in a provenance graph [13]. There are also applications for node and edge classification in biological networks [14].

3. **Community Detection**: The most famous early community detection application was to finding groups of members within a karate club that aligned with different leadership [15]. It is also used in the realm of politics, to detect ideologies in graphs of users or article citations [16]. Finally, community detection has applications to social networks such as clustering users based on the structure of the graph as well as the content of the topics that they are discussing in post data [17]

4. **Link Prediction**: Link prediction is the problem of finding new links between nodes in a graph. It is often used in recommender systems for e-commerce or social networks. In biology, link prediction is used to predict protein-protein interactions based on shared biological pathways [18].

1.2 Node Embeddings and Graph Representations

The objective of node embeddings is to find a representation of each node in a graph in d-dimensional space such that similar nodes in the graph will have embeddings that are close together in vector space. These representations are extremely adaptable for a wide variety of applications, especially due to their compatibility with traditional machine learning models. There are many potential ways to define similarity between nodes, so different methods of node embeddings are more practical for certain uses than others.

For instance, we could consider similarity between nodes based on their proximity in the graph. Nodes with this property will likely have coincidental nodes within their neighborhoods and (assuming the graph is undirected) will have similar routing times/distance to the other nodes in the graph. Alternatively, we could define similarity to be based on structural similarity. For exmaple, nodes that have a similar degree or neighborhoods from a structural perspective (such as being a hub or bridge, or appearing in a clique) may be considered similar.

The most popular node embeddings are general purpose, and applicable to a wide range of problems. However, there are embedding techniques that are optimized to solve specific problems. In this thesis we will see many examples of both types of node embeddings, and will compare their performance in certain domains.

The most common approaches to produce node embeddings include various random walk strategies, matrix factorization techniques, and deep learning approaches. Common random walk strategies include DeepWalk [19], Node2Vec [20], Struc2Vec [21], Walklets [22], and HARP [23], which define node representations based on co-occurrence in these walks. Node2Vec consistently performs well in my research, and I use it heavily in almost every project in this thesis. Node2Vec utilizes the Skip-Gram model to generate embeddings, which encodes pair associations as a function of the dot product. Matrix factorization techniques such as Graph Factorization [24], GraRep [25], HOPE, [26], and Prone [27] manipulate the graph adjacency matrix to more explicitly represent multi-level graph structure. GraRep for example computes and then merges k-step transition probability matrices of the node pairs to produce their final representations. Deep learning techniques are relatively new, the most famous being Graph Neural Networks (GNN) [28] which apply a neural network to neighborhood aggregations of node feature vectors. A famous deep learning technique for embeddings that we will explore further in this thesis is GraphSage [29].

1.3 Graph Theory and Machine Learning

Many techniques that traditionally were considered from a combinatorial or theoretical perspective have been attempted with machine learning due to its advantages in computability and performance. By applying machine learning algorithms to extract meaningful patterns from graph data, researchers can uncover hidden relationships, detect anomalies, identify community structures, and infer latent properties of networks. These insights contribute to a deeper understanding of complex systems and phenomena modeled as graphs, informing decision-making and guiding further analysis and exploration.

Machine learning algorithms have addressed scalability and efficiency challenges in graph theory research by providing methods that are successful for large-scale graph analysis. Graph algorithms and models leveraging machine learning techniques can efficiently handle massive graphs with massive amounts of of nodes and edges, enabling the analysis of complex network structures and interactions at a large scale. This is necessary for our modern digital age, where we often are working with a plethora of data.

The integration of machine learning and graph theory has spurred interdisciplinary research at the intersection of computer science, statistics, and other fields. Researchers are developing innovative methods and algorithms that combine techniques from machine learning, graph theory, and domain-specific knowledge to address complex challenges in diverse application domains such as bioinformatics, social sciences, healthcare, finance, and cybersecurity. These interdisciplinary approaches leverage the complementary strengths of machine learning and graph theory to tackle real-world problems and drive scientific and technological advancements.

1.4 Overview

This thesis aims to provide a comprehensive survey of common graph problems from a machine learning perspective. The topics covered include fundamental graph problems such as distance approximation, community detection, and cross-network alignment.

Each chapter represents a project focusing on machine learning for graph theory. These projects are unified by the theme of machine learning on graphs, graph embeddings, and representations of graphs:

- Distance Labeling
- Shortest Path Distance Approximation
- Distance Sensitivity Oracle
- Cross-Network Alignment
- Stochastic Block Model
- Geometric Wavelet Embeddings

In each chapter we will provide extra background information specific to the domain, outline a method and present experimental results and offer concluding thoughts. First, we will consider the problem of shortest path distance approximation from a machine learning perspective [30]. In the next chapter, we discuss our method to create labels optimized to be as small as possible for distance computation. Secondly, we introduce Adaptable Node Embeddings for Shortest Path Distance Approximation (ANEDA) [31], a lightweight technique for shortest path distance approximation that is optimized for simplicity and scalability. Then, we consider a machine learning approach to Distance Sensitivity Oracles (DSO) where we efficiently query replacement paths in failure-prone graphs, achieving near-optimal solutions with minimal recomputation overhead [32]. Collectively, these advancements contribute to enhancing the efficiency and scalability of distance approximation algorithms across various applications via machine learning.

The next project we will discuss is a deep learning approach to Cross-Network Alignment, the problem of aligning two separate networks with inherent overlap. In the following chapter, we will shift from representation learning to focus on community detection in a randomly generated graph model. Finally, we will head back to the representation space and go over Geometric wavelet embeddings on graphs, a novel technique for reducing the dimension of graph embeddings for various graph tasks.

In total, these projects tell a story about how machine learning and specifically node represenations and learned embeddings can be utilized to solve complex problems over diverse types of networks and domains.

Chapter 2

Distance Labeling

2.1 Abstract

In this chapter we discuss a novel learning-based approach to distance labeling inspired by collaborative filtering. We demonstrate that it is able to outperform theoretical baselines in minimizing label size and demonstrates promising results in practical scenarios with real-world graphs.

2.2 Background

2.2.1 Problem Definition

Given an undirected arbitrary graph G_i with length i and any two nodes u, v in G_i. Denote the distance between u, v in G_i by $d_{G_i}(u, v)$, the distance labelling problem aims to find a labeling scheme $\ell(\cdot)$ and a function f such that

$$f(\ell(u), \ell(v)) = d_{G_i}(u, v)$$

For Approximate Distance Labeling, denote the number of bits required to store the label of each node by r_l, the target is to find a labeling scheme $\ell(\cdot)$ and a function f such that

$$\text{minimize} \quad |f(\ell(u), \ell(v)) - d_{C_i}(u, v)| \quad \text{subject to} \quad r_l \leq M$$

where M is some pre-defined positive value.

2.2.2 Overview and Related Work

In this chapter, we will focus on approximate distance labeling with the goal of using as few bits as possible for storing labels while achieving satisfactory distance approximation results. Via the utilization of machine learning techniques, we enhance the efficiency of data representation, enabling a higher data-to-bit ratio compared to conventional labeling methods. This allows for encoding richer data that machines can interpret, as opposed to the more limited information typically interpretable by humans in traditional labeling approaches. Allowing for time expensive pre-processing in the form of training and some approximation error as a trade-off, we can achieve fast time distance querying in large graphs with fewer bits.

Distance labeling was investigated from the theoretical perspective for various types of graph classes, e.g., trees [33, 34, 35], planar graphs [36], sparse graphs [37] and general graphs [38].

Common approaches to produce an embedding map for graphs include random walk strategies such as DeepWalk [19] or Node2Vec [20] and matrix factorization techniques such as GraRep [25]. These embedding techniques are famous and general, demonstrating high performance on various graph tasks such as clustering or classification.

In response, several deep learning architectures have been proposed to train on graph embedding inputs as a strategy for distance approximation [39], [40], [41]. These approaches use an existing graph embedding technique for labels ℓ and then try to find an optimal measure f using deep learning, or train both ℓ and f jointly.

Among the first to apply graph embeddings to the shortest paths problem was Orion [42]. Inspired by the successes of virtual coordinate systems, a landmark labelling approach was employed, where positions of all nodes were chosen based on their relative distances to a fixed number of landmarks. Using the Simplex Downhill algorithm, representations were found in a Euclidean coordinate space, allowing constant time distance calculations and producing mean relative error (MRE) between 15% - 20% [42]. Other existing coordinate systems have also been used. Building off of network routing schemes in hyperbolic spaces, Rigel used a hyperbolic graph coordinate system to reduce the MRE to 9% and found that the hyperbolic space performed empirically better across distortion metrics than Euclidean and spherical coordinate systems [43, 44]. In road networks, geographical coordinates have been utilized with a multi-layer perceptron to predict distances between locations with 9% MRE [45].

In addition to these coordinate systems, general graph embedding techniques have recently been employed to handle shortest path queries to great success. In 2018, researchers from the University of Passau proposed node2vec-Sg[39]. To find the shortest path between nodes s and t, their Node2vec and Poincare embeddings were combined through various binary operations and fed into a feed-forward neural network, which was trained only on the distances between l landmark nodes $l << n$ and the rest of the graph. The model which took concatenated Node2vec embeddings performed the best, with an MRE between 3% to 7% .

Researchers have also demonstrated the accuracy of graph embeddings learned alongside distance predictors, to produce representations more specific to the shortest path task. Vdist2vec directly learned vertex embeddings by passing the gradient from the distance predictor back to a $N \times k$ matrix, achieving an MRE between 1% to 7% [46]. Huang et al. computed shortest path distances on road networks using a hierarchical embedding model and achieved an MRE of 0.7% [8]. Most recently, ndist2vec built upon the landmark learn-

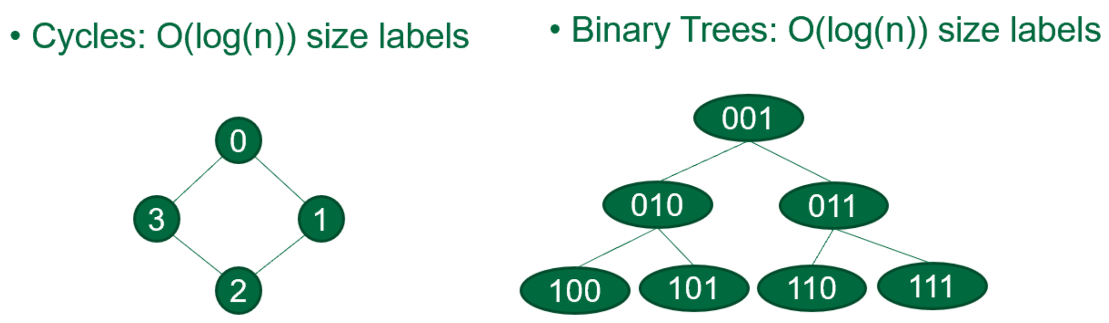

• Cycles: O(log(n)) size labels
0
3
1
2
• Binary Trees: O(log(n)) size labels
001
010
011
100
101
110
111

Figure 2.1: O(log(n)) labels of cycles and trees, which allow for perfect distance reconstruction using only pairs of labels.

ing, graph embedding, and neural network aspects of all of these approaches, reporting an MRE of 3.4% with a dataset on the order of $O(n)$.

Current works for estimating the shortest path lengths between two nodes are limited by the representations they learn. They rely on datasets which, even using schemas like landmark labelling or hierarchical, are proportional to n, the number of nodes in the network[47, 8]. This presents a significant bottleneck for larger graphs.

2.3 Datasets

We utilized synthetic datasets for cycles and trees, as well as real-world graphs obtained from [48] for our experiments.

For cycles, we generated a dataset $D_N = \{(u, v, d) : u, v \in C_i, 3 \leq i \leq L\}$ for experiments. Here, $G = \bigcup_i C_i$ represents a union of disjoint cycles, where C_i is a cycle. V_{C_i} is the set of all nodes that belong to C_i, then $V = \bigcup_i V_{C_i}$ is the set of nodes that exist in any cycle within G.

For trees, we followed a similar procedure to generate a dataset $D_N = \{(u, v, d) : u, v \in T_i, 3 \leq i \leq L\}$ for experiments. Define $G = \bigcup_i T_i$ be a union of disjoint random trees, V_{C_i} be the set of all nodes that belong to C_i, then $V = \bigcup_i V_{C_i}$ is the set of nodes that exist in any tree that belongs to G. In both the trees and cycles dataset, each node is assigned a unique integer id.

The real world graphs dataset comprises enzymes graphs obtained from [48] which provides the graph edge lists. We processed the data to ensure that every node in the entire set has a unique id and that there are no gaps between the ids of any two nodes. The dataset $D_N = \{(u, v, d) : u, v \in G_i, 3 \leq i \leq L\}$ was then created from this collection of real-world graphs, where G_i is a graph in the collection. We specifically use the chem-ENZYMES-g1 and chem-ENZYMES-g118 graphs.

2.4 Method Summary

We primarily employed synthetic datasets consisting of cycles and trees for this project, although we also tested on real-world networks to determine how well our model generalized. For the models, we used deep learning for both the map producing the labels and the measure, which takes in two labels concatenated and produces a distance approximation. The map is a pytorch embedding layer, and the measure is a two layer collaborative filtering model, motivated by the fact that nodes that are close together are likely to have similar neighborhoods. We controlled quantization with two methods, one which quantized labels during training, and one which quantized labels post training.

We initialize the feature of the node pairs as the concatenation of their respective quantized labels. After processing this feature through two layers of our model, an output prediction $\hat{Y}$ of the distance between two nodes is produced. Each layer is followed by an activation function. Let n denote the number of samples; we have $X = \{(l(u), l(v)) : u, v \in C_i \in G\}$, $X \subseteq R^{n \times 2}$. The training phase can be defined formally as:

$$\hat{Y} = \sigma((ReLU(XW_1))W_2)$$

where $W_1 \subseteq R^{2 \times d}$, $W_2 \subseteq R^{d \times 1}$ are the weight matrices to be trained. The final activation function σ is modified so that the prediction will be re-ranged to fit the dataset's distance range.

2.4.1 Label Quantization Strategies

In our work, we employed two distinct quantization strategies: training quantization and post-training quantization.

Training quantization is implemented concurrently with the learning process. Once the labels are generated, we quantize them to further reduce their size. This process takes place during the forward training pass, where we convert the node embeddings into a binary form by taking the sign of each embedding value. The size of each node's embedding is determined by the quantization level, and by manipulating the number of bits i, we can control the trade-off between the label size and the accuracy of node pair distance computation. Despite its intuitive appeal, this approach exhibited suboptimal performance on cycles and graphs, while showing improved results for trees.

On the other hand, post-training quantization is applied after the training phase. In this strategy, we first complete the training without any quantization. Afterward, we quantize the resultant embeddings, reintroduce these quantized labels into the model, and execute a forward pass to derive the predictions. This method demonstrated superior efficacy for cycles and graphs, but was less effective when applied to random trees.

We hypothesize that this disparity in performance is likely related to the prevalence of cycles in the graph datasets, particularly the high frequency of triangles. Thus, it is important selecting the appropriate quantization method based on the characteristics of the graph structure.

2.5 Experimental Results

We use the Mean Squared Error(MSE) and Mean Relative Error(MRE) as our metrics for experimental results. Given the prediction $\hat{Y} = \{\hat{y}_i : 1 \leq i \leq N\}$ and labels $Y = \{y_i :$

$1 \leq i \leq N\}$, the MSE and MRE are computed as follows:

$$MSE(\hat{Y}, Y) = \frac{1}{N}(\hat{Y} - Y)^T(\hat{Y} - Y)$$

$$MRE(\hat{Y}, Y) = \frac{1}{N}\sum_{i=1}^{N}\frac{|\hat{y}_i - y_i|}{|y_i|}$$

We define parameter α as a float that ranges from 0.0 to 1.0, the final loss l is computed as the combination of MSE and MRE:

$$l_\alpha(\hat{Y}, Y) = \alpha MSE(\hat{Y}, Y) + (1 - \alpha)MRE(\hat{Y}, Y)$$

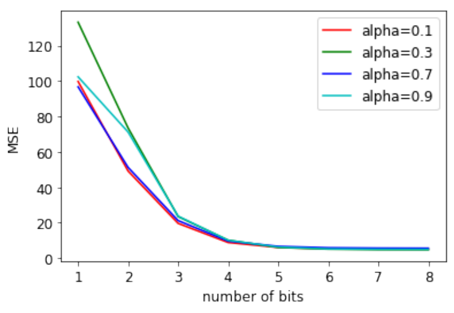
We found that an α value of 0.5 produces lower loss values in the case of cycles, as seen in Table 2.1. For the following experiments, we used an alpha value of 0.5.

Table 2.1: Results with D_6

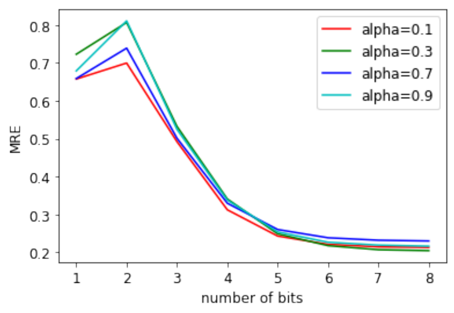
	$\alpha = 1$		$\alpha = 0$		$\alpha = 0.5$	
	MSE	**MRE**	**MSE**	**MRE**	**MSE**	**MRE**
1	149.175964	0.798590	525.602400	1.223840	97.190900	0.648095
2	63.960815	0.622632	90.706116	0.876957	57.236360	0.748088
3	32.998089	0.605301	30.461815	0.582474	21.510620	0.509291
4	12.950354	0.372077	17.285600	0.379353	9.796305	0.325589
5	6.635018	0.263844	13.536446	0.292693	6.938656	0.262967
6	4.4964622	0.227134	12.649083	0.259946	6.095809	0.241920
7	4.551461	0.215136	12.409961	0.249289	5.879529	0.235888
8	4.432967	0.212821	12.357460	0.246148	5.845781	0.234776
9	4.405716	0.211925	12.345035	0.244986	5.829654	0.234216
10	4.399604	0.211756	12.338661	0.244760	5.827143	0.234158

For cycles, our method beat the standard combinatorial baseline with respect to label size by 33% with a trade-off of .22 Mean Relative Error (MRE), achieving satisfying predictions while utilizing significantly less bits. If we allow for an MRE < 0.4, we can achieve a 60% reduction in bits.

However, it produces generally worse performance on trees compared to cycles. The

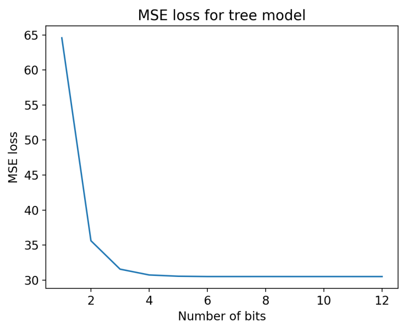


(a) MSE using different number of bits

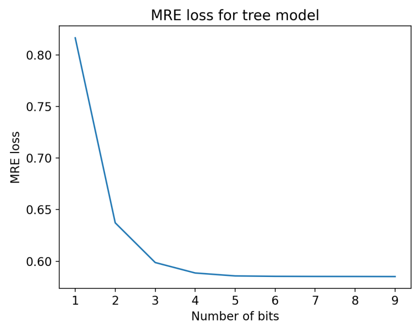


(b) MRE using different number of bits

Figure 2.2: Model's performance on Cycles using different number of bits



(a) MSE using different number of bits

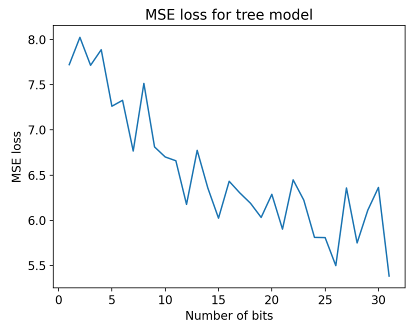


(b) MRE using different number of bits

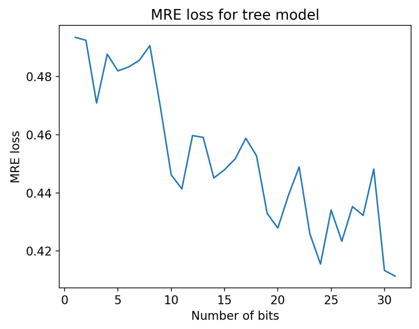
Figure 2.3: Model's performance on Trees using different number of bits with post-training quantization

model's performance on trees plateaus at around 3 bits with post-training quantization, which is an improvement over the theoretical bound of 11, but there is a not insignificant error of .6 MRE/ 32 MSE associated with this label size. Allowing additional bits does not improve model performance significantly. With the training quantization method, extra bits do seem to trend error downwards, however the effect on error is less dramatic than the results on cycles. Trees require a significantly higher number of bits than cycles to achieve an MRE around .45.

After fine-tuning our model on both tree and graph datasets, we proceeded to evaluate its performance on general graphs. Notably, employing the post-training quantization methodology yielded the best results experimentally. These results on general, real world graphs are promising and demonstrate that this method has potential viability for real world



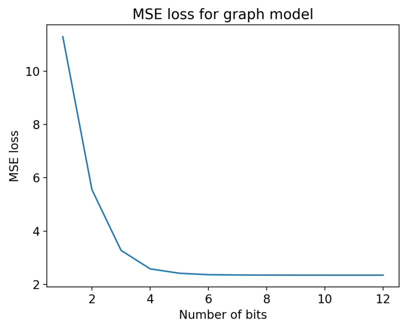
(a) MSE using different number of bits



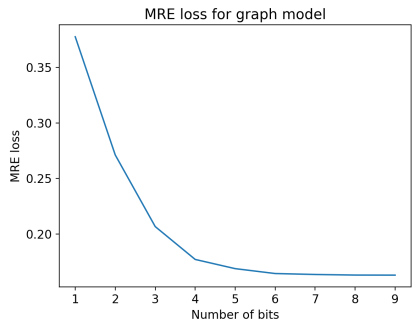
(b) MRE using different number of bits

Figure 2.4: Model's performance on Trees using different number of bits with quantization during training

applications.



(a) MSE using different number of bits



(b) MRE using different number of bits

Figure 2.5: Model's performance on General graphs using different number of bits

2.6 Conclusion

In this chapter, we have presented a novel learning-based approach to the Distance Labeling problem, aiming to reduce the size of labels required. Our approach demonstrates substantial improvements over existing combinatorial and approximation baselines, offering increased efficiency and reduced storage needs. Looking forward, an intriguing avenue for future research lies in extending our methodology to accommodate larger and more complex graph structures, including scenarios with node failures. Additionally, we are interested in exploring online cases where cycles are presented sequentially. This direction promises to further expand the applicability and impact of our approach in the field of algorithmic graph theory.

Chapter 3

Shortest Path Distance Approximation

3.1 Abstract

Shortest path distance approximation is a crucial aspect of many graph algorithms, in particular the heuristic-based routing algorithms that make fast, scalable map navigation possible. Past literature has introduced deep learning models which try to approximate these distances by training on graph embeddings. We propose a more lightweight technique than the embedding and graph neural network scheme, which involves training the embeddings directly, using either previous embedding techniques or geographic coordinates as a good initialization. We demonstrate applications to deep A* routing and learned road maps. Through experiments on several road and social networks, we show our model's error reduction of up to 75% against two recent deep learning approaches, and its competitive performance against the larger, state-of-the-art architecture.

3.2 Background

While the previous chapter's project attempted to minimize the size of the labels for the distance labeling problem, our second approach ANEDA (Adaptable Node Embeddings for Shortest Path Distance Approximation) seeks to create a simpler, more lightweight model for distance labeling with human interpretable embeddings for labels. We explicitly choose a function M for the measure of the distance based on label vectors ℓ, initialize $\ell = \ell_0$ from another graph embedding (not necessarily explicitly trained for distance approximation) and then try to find an optimal ℓ through training a lightweight model. We then tune this initialization in order to get improved shortest path approximations with respect to a chosen distance measure, such as the Euclidean distance or dot product.

Besides comparable performance to state-of-the-art techniques, the merits of this approach are:

1. **Adaptability**: Our model can be used to find node representations in a particular space by tweaking the distance measure. This could be useful in order to create distance-preserving graph embeddings in spaces with desired properties, or tailor representation for graphs with known or theorized properties. The model can also be easily exported as a 2d array and queried for downstream tasks.

2. **Size**: Our model is an embedding matrix of size $|V| \times D$, whereas an approach like Vdist2vec [40] contains an embedding layer of the same size as part of its larger model. This smaller size is likely to improve train and evaluation time. Our model can also optionally be initialized with another embedding of size $|V| \times k$, where $k \leq D$, that can be discarded after initialization. Vdist2vec on the other hand requires a one-hot input of size $2|V|$ per example to train.

3. **Simplicity**: Our model requires only an embedding matrix and trainer (we use node2vec as our base code), plus a simple distance measure in order to implement the model for a specific task.

	|V|	|E|	AVG DEG	d_{max}
HARTFORD (**HF**)	1581	2470	2.75	12 KM
SURAT (**SU**)	2591	3670	2.86	51 KM
EGO-FACEBOOK (**FB**)	4039	88234	2	8
DONGGUAN (**DG**)	8315	11128	2.32	160 KM

Table 3.1: Descriptions of datasets used in our experiments

3.3 Datasets

For datasets, we used road networks of Santa Ana CA [49], Hartford CT [49], Dongguan China [50], and Surat, India [50] as well as a facebook social media network [51].

3.4 Method Summary

In order to make the model as lightweight as possible, we developed a simple two layer multi-layer perceptron (MLP) model for ℓ. In order to improve training time and potentially performance, we use three embedding types as initializations for the model: Node2Vec [20], GraRep [25], and geographic location embeddings. We also modified the loss function based on fixed metric M, of which we considered a variety of options including including common norms for Euclidean geometry and measures for hyperbolic and elliptical geometry.

We chose Node2Vec and GraRep as initial embeddings because they are both general purpose embedding techniques that are highly accessible and have been proven to provide significant performance in experimental settings for a variety of contexts, including distance approximation tasks. The choice of a multi-layer perceptron embedding model was motivated by the fact that it is lightweight and fast. By initializing the embeddings with Node2Vec or GraRep, the idea is to speed up the training time of the neural network by starting at a meaningful representation of the graph rather than from random weights.

3.4.1 Data Processing

In order to produce node pairs with distance labels for the model to train with, we follow a similar approach to [39], which is to randomly select l nodes from the graph to use as landmarks, and compute the distance between this landmark and all other nodes using Dijkstra's. Given landmark u and other node v, each training example in our dataset is simply $< u, v, d_{u,v} >$. Since we are tuning the embeddings directly, we only need the node indices as inputs. Our embedding is instead introduced when evaluating the loss, similar to Node2Vec [20], whereas the approach in [39] may need to store the full vector input since it is a function of the node embeddings for u and v. This greatly reduces the dataset space requirements of our approach, since each example requires storing 3 values instead of $D + 1$ values.

Another notable difference is [39] makes their validation sets include entirely distinct nodes from the test set. We cannot do this for our model since it would prevent embeddings for the validation nodes from ever being trained, so we instead remove train landmarks from the validation sets to prevent the two sets from having any of the same pairs. As suggested by [41], for weighted graphs we also normalize the edge weights by dividing by the largest across all edges before computing the distance labels.

3.4.2 Initialization

In order to improve training time and potentially performance, we use two embedding types as initializations for the model.

For any graph, we can train graph embeddings not designed specifically for distance approximation, such as Node2Vec, and then tune our model to produce embeddings that approximate graph distances very well. We test specifically with Node2Vec [52] and GraRep [25] embeddings.

If the graph is geographic, we can also use geographic coordinates of each node as an initialization. We evaluate several geographic networks with either latitude-longitude coordinates or UTM coordinates of the nodes included. For latitude-longitude coordinates, where $\varphi \in [-\frac{\pi}{2}, \frac{\pi}{2}]$ = latitude and $\lambda \in [-\pi, \pi]$ = longitude, in general, we convert to 3-dimensional Cartesian coordinates using the identity

$$x = \cos(\varphi)\cos(\lambda), \quad y = \cos(\varphi)\sin(\lambda), \quad z = \sin(\varphi)$$

This is done since differences in latitude and longitude reflect distances differently depending on location, and the change also results in coordinates on the same scale as the normalized graph distances. No changes are made to UTM coordinates other than normalization since these have a pre-applied projection into 2-dimensional Euclidean space.

We make use of k-dimensional coordinate vectors when available to initialize the first k of our D-dimensional embeddings. The remaining $D - k$ dimensions are initialized at random from a normal distribution with a small variance of $1/D$. This allows us to train embeddings of a much larger size than the 2 or 3 dimensional geographic coordinates, or add additional dimensions to initial embeddings for tuning. In general, though, for trainable graph embeddings (like Node2Vec) we use $D = k$.

The different initialization techniques also offer the opportunity to choose distance measures for training with an expected relationship to the initial embedding. For instance, when using the Euclidean distance for training and latitude-longitude initialization (transformed as mentioned above), the initial distance measure is an underapproximation (although very accurate for small sections of the Earth like cities) of the geographic distance between the nodes. This measure is a reasonable baseline approximation for downstream tasks like routing, as we observe in later experiments. In Figure 3.1, we visualize how these initial coordinates are projected by the model in a graph where the geographic distance does not perform adequately. This is done by applying the transformation from Lat-Long to Cartesian for the initialization, applying the reverse on the output embeddings and then modifying the graph coordinate attributes. The Santa Ana, CA graph from OSMnx has two "islands" which, except for several bridging edges, are separated by highways that extend outside the city boundaries. The model moves these islands up and farther left and right, while also condensing them. Nodes in regions with many short edges are generally condensed, while holes in the graph are created in areas with fewer, longer streets. This model was trained with $D = k = 3$, so this visualization may not fully reflect the representations learned in embeddings with many additional dimensions.

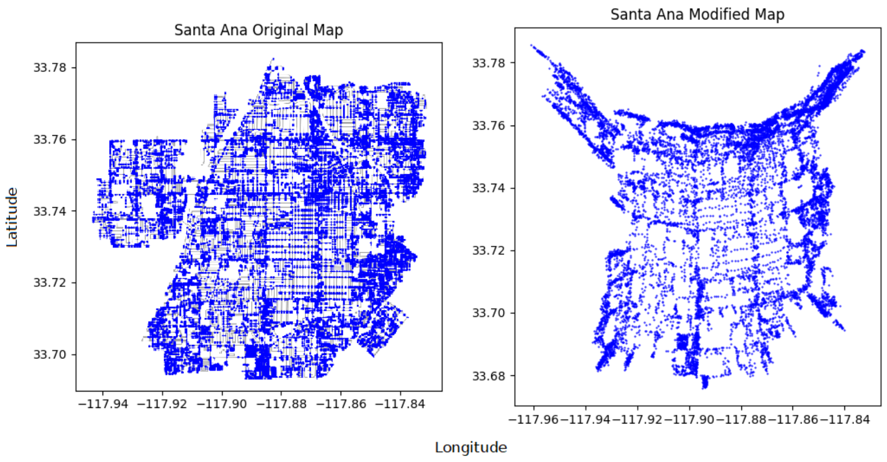


Figure 3.1: 2D ANEDA embeddings generated from geographic location initialization, visually showing distance preserving modifications to the original labels

3.4.3 Distance Measures

We tested several different distance measures to train the embeddings on, including common norms for Euclidean geometry and measures for hyperbolic and elliptical geometry.

p-norm

Using several different p values, we tested the p-norm of the difference between two node embeddings which is given by

$$\hat{d}_{u,v} = \left(\sum_{i=1}^{D} (\mathbf{u}_i - \mathbf{v}_i)^p \right)^{1/p}.$$

Poincare Hyperbolic

We additionally use two measures of distance in hyperbolic geometry based on the Poincare Disk and Minkowski Hyperboloid models for hyperbolic space.

In the 2d Poincare Disk model, points are fixed inside the unit disk, with the edge of the disk representing infinite distance. As mentioned in [53], we can extend this idea to the D-dimensional unit ball and then the distance measure is

$$\hat{d}_{u,v}(u,v) = \text{arcosh}\left(1 + 2\frac{||\mathbf{u}-\mathbf{v}||^2}{(1-||\mathbf{u}||^2)(1-||\mathbf{v}||^2)}\right).$$

Minkowski Hyperbolic

In the Minkowski Hyperboloid model, points are fixed inside the forward sheet of a two sheeted hyperboloid in $D+1$ dimensional space. All points $\mathbf{x} \in \mathbb{R}^{D+1}$ must satisfy

$$\mathbf{x}_0^2 - \mathbf{x}_1^2 - \cdots - \mathbf{x}_D^2 = 1.$$

Then the distance measure between 2 points $u, v \in \mathbb{R}^{D+1}$ is

$$\hat{d}_{u,v}(u,v) = \text{arccosh}(\mathbf{u}_0\mathbf{v}_0 - \mathbf{u}_1\mathbf{v}_1 - \cdots - \mathbf{u}_D\mathbf{v}_D).$$

In practice, in order to enforce the constraint for each point $\mathbf{x}$, we keep its values in the last D dimensions and set the first dimension, x_0 to:

$$x_0 = \sqrt{||\mathbf{x}||^2 + 1}$$

The measure then becomes

$$\hat{d}_{u,v}(u,v) = \cosh^{-1}\left(\sqrt{(||\mathbf{u}||^2+1)(||\mathbf{v}||^2+1)} - \mathbf{u}\cdot\mathbf{v}\right).$$

In practice, in order to enforce the constraint, on each distance measure query we keep $\mathbf{x}$ D-dimensional and set the x_0 coordinate:

$$x_0 = \sqrt{||\mathbf{x}||^2 + 1}$$

so the measure becomes

$$\hat{d}_{u,v}(u,v) = \cosh^{-1}\left(\sqrt{(||\mathbf{u}||^2 + 1)(||\mathbf{v}||^2 + 1)} - \mathbf{u} \cdot \mathbf{v}\right).$$

Elliptic

We also tested using measures in a positive curvature geometry since this is the true geometry of geographic points. Assuming a radius of 1, the elliptical distance is simply the angle between the points in radians, which can be obtained from the dot product:

$$\hat{d}_{u,v} = \cos^{-1}\left(\frac{\mathbf{u} \cdot \mathbf{v}}{||\mathbf{u}||||\mathbf{v}||}\right).$$

A natural initialization to use with the elliptic distance is geographic coordinates (still with the Cartesian transformation), since this gives the geographic distance exactly, and so should in theory provide a better initialization than with the Euclidean distance measure.

Inverse Dot

With inspiration from the elliptic distance, we tried several other functions besides inverse cosine which would take the normalized dot product δ and "invert" it by mapping $\delta = -1$ to a large distance and $\delta = 1$ to 0. We found the most effective to be

$$\hat{d}_{u,v} = \left(1 - \frac{\mathbf{u} \cdot \mathbf{v}}{||\mathbf{u}||||\mathbf{v}||}\right) * d_{max}/2$$

where d_{max} is the graph's diameter. In practice, if d_{max} is unknown for a large graph it can estimated, as long as the estimate is greater than d_{max}. We need this so that the measure is still able to predict d_{max} despite the restriction of δ to $[-1, 1]$.

3.5 Experimental Results

We compared our results to that of Brunner et. al. [41], Rizi et. al. [39], and Qi et. al. [40] and found improved results for some combinations of initializations and metrics. For most experiments, we used a D value of 128, resulting in much improved performance compared to our original distance labeling project which was aiming to minimize bits. Overall, we found that the ANEDA model was often able to beat other state of the art methods for distance approximation, most commonly with Node2Vec initialization with L6 norm or inverse dot measure.

3.5.1 Road Network Experiments - Vdist2Vec

In Table 3.2 and Table 3.3, we show results against Vdist2vec on two geographic networks: **SU** and **DG** from [40]. We observe that at least in geographic networks, our Inv-Dot model is competitive with Vdist2vec but does not surpass it entirely in error as we will see with the other experiments. On **SU**, Inv-Dot performs between the initial Vdist2vec and Vdist2vec-S in terms of MRE regardless of initialization. When additionally trained on MAE, our model surpasses the base Vdist2vec entirely. On **DG**, our Inv-Dot with node2vec initialization model achieves state-of-the-art in terms of MRE, which occurs whether we train with MAE or MRE as the loss.

3.5.2 Road Network Experiments - Brunner

Against Brunner we find that with coordinate initialization several of our measures are competitive with their graph neural network architecture, as shown in Table 3.5. L4, L6,

Table 3.2: Experiment A - Surat

Technique	Initialization	Performance		Params
		MAE	MRE	
L6-Norm	Random	0.0206	0.0266	lr=0.001
	Coord	0.0328	0.034	
	node2vec	0.0194	0.0261	
Inv-Dot	Random	0.0161	0.0197	
	Coord	0.0163	0.0207	
	node2vec	0.0159	0.019	
	GraRep	0.0164	0.0209	
Inv-Dot MAE	node2vec	0.0109	0.0239	
Inv-Dot MSE	node2vec	0.0098	0.027	lr=0.0003
Vdist2vec base	N/A	0.0113	0.027	
Vdist2vec-S		**0.0067**	**0.014**	

Table 3.3: Experiment A - Dongguan

Technique	Initialization	Performance		Params
		MAE	MRE	
L6-Norm	Random	0.0355	0.0286	lr=0.001
	Coord	0.0679	0.0374	
	node2vec	0.0349	0.0304	
Inv-Dot	Random	0.0207	0.0345	
	Coord	0.021	0.0121	
	node2vec	0.0196	**0.01**	
Inv-Dot MAE	node2vec	0.0178	0.0123	
Inv-Dot MSE	node2vec	0.0211	0.0171	
Vdist2vec base		0.0118	0.015	
Vdist2vec-S		**0.0062**	0.014	

and L8 are all within the same range of performance as these networks, illustrating that rich representations can be learned even when the distance measure is not an ideal approximation with respect to the provided initial coordinates. The more typical L2 distance also performs in the range in MAE, even though we train on MRE. As expected, hyperbolic distance models do not perform as well as p-norms or dot product, which are more intuitive measures for graphs representing physical geometries. As with the previous experiment on geographic networks, our best performance comes from the Inv-Dot measure, which surpasses all comparison models on this experiment.

Since Qi et al. and Brunner both ran experiments on the **SU** graph, we can also get some sense of how the difference in experiment setup affects the performance of our methods. We found the Brunner setup to be much more stable generally and so required no hyperparameters tweaks for specific models. In comparison to the Qi experiment, it used a lower learning rate (0.01 $\rightarrow$ 0.001), smaller batch size (2500 $\rightarrow$ 512) and more train epochs (20 $\rightarrow$ 100). Most significantly, Qi et al. train on all node pairs, whereas the Brunner experiments use 10% train ratio and tested on the remainder. We can see that between these two experiments L6 and Inv-Dot setups performed similarly. This is good news since it suggests that the models achieve good representations without simply memorizing distances for the provided node pairs. In the future, experiments with lower train ratios and larger graphs may be more useful to evaluate performance, since this line of research in distance approximations with embeddings or neural networks is only useful when $|V|$ is too large to be able to generate the shortest path distances for all node pairs in a feasible amount of time with modest space available.

3.5.3 Social Network Experiments

In the non-geographic setting, we show a significant performance increase over the Rizi et al. model as evidenced by Table 3.4 showing approximation errors on the ego-facebook-original graph. Two of our models, elliptic and L6 norm with node2vec, show serious

Table 3.4: Experiment B - Facebook

Technique	**Initialization**	**Performance**		**Params**
		MAE	**MRE**	
L2-Norm	Random	0.114	0.0586	lr=0.008
L6-Norm	Random	0.0482	0.022	lr=0.001
	node2vec	**0.0267**	**0.0143**	lr=0.03
	GraRep	0.1317	0.056	
Elliptic	Random	0.0272	0.0157	lr=0.0003
Poincare	Random			
Minkowski	Random	0.4068	0.1464	lr=0.005
Inv-Dot	Random	0.6278	0.1637	lr=2e-5
	node2vec	0.5001	0.1365	
Rizi	node2vec $\oslash$	0.118	0.038	
Rizi	node2vec $\ominus$	0.197	0.071	

Table 3.5: Experiment C - Surat

Technique	**Performance**	
	MAE	**MRE**
L2-Norm	0.0411	0.0605
L4-Norm	0.0299	0.0385
L6-Norm	0.0291	0.0357
L8-Norm	0.0307	0.0372
Poincare	0.0951	0.0404
Minkowski	0.3525	0.2101
Inv-Dot	**0.0161**	**0.0229**
CN	0.0265	0.0378
SCN	0.0238	0.0309
SECN	0.0433	0.0450
SACN	0.0419	0.0436

Table 3.6: Hartford Routing Results
(Q = percent unnecessary visits)

Technique	**Initialization**	**Q-Average**	**Q-50th**	**Q-90th**	**Q-99th**	**Routing Time**
L2-Norm	Coord	14.45	8.7	37.5	86.06	**18:25**
L6-Norm	Coord	20.09	8.33	67.86	97.14	22:21
	node2vec	17.12	7.55	54.05	96.16	21:58
Inv-Dot	Coord	22.55	13.89	60.98	89.77	20:27
	node2vec	**12.53**	**7.14**	**32.56**	**82.59**	18:44
Poincare	Coord	15.86	9.76	40.32	86.23	18:32
Spherical	Coord	15.33	9.43	38.36	85.71	19:01
Geographic Dist		23.86	8.33	54.29	96.81	22:54

performance jumps relative to the Rizi et al. model. We display their result with the best (averaging, denoted $\oslash$) or second best (subtracting, denoted $\ominus$) performing input vector merging scheme for this graph. We should note that the elliptic measure result could be considered more significant since it was achieved after lowering, rather than raising the learning rate from the default for the experiment. Tweaking the learning rate down was helpful for most models, which is perhaps related to the low number of epochs set by Rizi et al. for this experiment.

3.5.4 Heuristic Routing Experiments

We also tested the distance approximations from ANEDA as heuristics for the A^* algorithm, to determine if they reduce the time and space to find the optimal route using the heuristic of geographic distance as a baseline. We found that Node2Vec initialization with inverse dot measure indeed provided improved distance approximations for A^*, resulting in significantly fewer unnecessary explored nodes, as seen in Table 3.6.

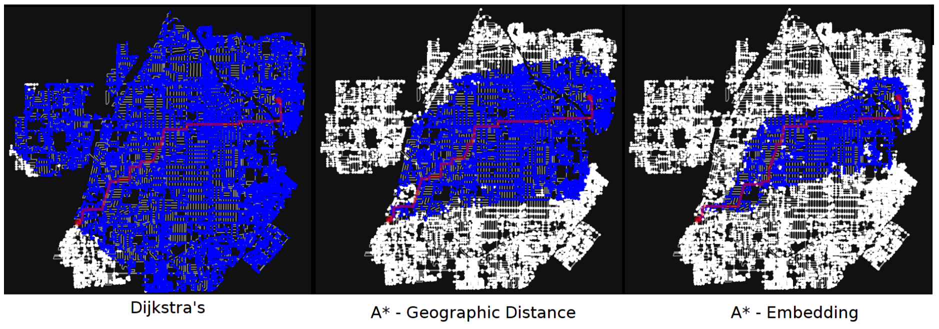


Figure 3.2: Santa Ana, CA routing visits with different A* heuristics

3.6 Conclusion

We've proposed a simple embedding approach, ANEDA, to learn distance preserving graph embeddings in explicit representation spaces, and demonstrated comparable or better performance against state-of-the-art deep learning architectures.

For future investigation, we consider the following areas of extension or improvement of our work:

1. Verify whether using specific measures is successful in most cases at embedding into the respective mathematical space (within some margin of error).

2. Modify measures for certain geometries (e.g. hyperbolic, elliptic) to produce better gradients.

3. Introduce a loss or training constraint to better enforce the triangle inequality so that our model defines a metric space.

4. Design a custom loss to target worst-case distance approximation, particularly for use in downstream routing tasks.

5. Apply the technique in an online learning setting, where embeddings must be updated as changes to the graph occur.

Chapter 4

Distance Sensitivity Oracle

4.1 Abstract

Our approach for Distance Sensitivity Oracles (DSO) was the first to use deep learning for re-rerouting paths after a node failure. We consider the simplest case where one node fails, and query the model for an updated shortest path without explicitly recomputing it. We reduced this problem to finding a pivot node to route through instead of the failed node. The performance we were able to achieve with this method was a significant improvement over any other state-of-the-art DSO approach.

4.2 Background

One of the most fundamental graph problems is finding a shortest path from a source to a target node. Efficient algorithms are known it becomes significantly harder as soon as parts of the graph are susceptible to failure. Although one can recompute a shortest replacement path after every outage, this is rather inefficient both in time and/or storage.

One way to overcome this problem is to shift computational burden from the queries into a pre-processing step, where a data structure is computed that allows for fast querying of replacement paths, typically referred to as a *Distance Sensitivity Oracle (DSO)*.

Such a DSO can be constructed, by utilizing the fact that the shortest path in a graph with failures consists of a concatenation of shortest paths in the original graph. That is, one can quickly compute a replacement path by determining a suitable pivot and concatenating the shortest paths from source to pivot and from pivot to target.

Unfortunately, finding such a pivot is far from trivial. In this project, we utilize node2vec, graph attention networks, and multi-layer perceptrons to find pivots such that the lengths of the resulting paths are close to that of the shortest replacement paths. More precisely, we evaluate our technique on a collection of real-world networks and observe that on average the computed replacement paths are longer by merely a few percentages compared to the optimal solution.

While DSOs have been extensively studied in the theoretical computer science community, to the best of our knowledge this is the first work to construct DSOs using deep learning techniques. In particular, in this work we first observe the existence of a combinatorial structure and then build on top of it a deep learning algorithm that utilizes the observed combinatorial structure. As mentioned above - we utilize the combinatorial structure of replacement paths as a concatenation of shortest paths and use deep learning to find the pivot nodes for stitching shortest paths into replacement paths.

4.2.1 Related Work

The problem of constructing a DSO is well-studied in the theoretical computer science community. Demetrescu *et al.* [54] showed that given a graph $G = (V, E)$, there is a DSO which occupies $O(n^2 \log n)$ space, and can answer a query in constant time. The preprocessing of this DSO, that is the time it takes to construct this DSO, is $O(mn^2 + n^3 \log n)$.

Several theoretical results attempted to improve the preprocessing time required by the DSO. Bernstein and Karger [55] improved this time bound to $\tilde{O}(mn)$ where $\tilde{O}(f)$ denotes $O(f \cdot polylog(n))$.

Note that the All-Pairs Shortest Paths (APSP) problem, which only asks the distances between each pair of vertices u, v, is conjectured to require $mn^{1-o(1)}$ time to solve [56]. Since we can solve the APSP problem by using a DSO, by querying it with (s, t, emptyset) for every s, t, the preprocessing time $\tilde{O}(mn)$ is theoretically asymptotically optimal in this sense, up to a polylogarithmic factor (note that, in practice, such polylogarithmic factors may be very large). Several additional results improved upon the theoretical preprocessing time by using fast matrix multiplication [57, 58, 59].

With respect to the size of the oracle, Duan and Zhang [60] improved the space complexity of [54] to $O(n^2)$, which is from a theoretical perspective asymptotically optimal for dense graphs $(i.e., m = \Theta(n^2))$. To do so, Duan and Zhang store multiple data-structures, which is reasonable for a theoretical work, however from a practical perspective the hidden constant is large. Therefore, it may also be interesting to consider DSOs with smaller space, at the cost of an approximate answer.

Here are several DSOs that provide tradeoffs between the size of DSO and the stretch (the length reported divided by the actual length):

- The DSO described in [61], for every parameter $\epsilon > 0$ and integer $k \geq 1$ has stretch $(2k-1)(1+\epsilon)$ and size $O(k^5 n^{1+1/k} \log^3 n/\epsilon^4)$.
- The DSO described in [62], for every integer parameter $k \geq 1$ has stretch $(16k-4)$ and size $O(kn^{1+1/k} \log n)$.

Note that even though the size of the above two DSOs for $k \geq 2$ is asymptotically smaller than $O(n^2)$, the stretch guarantee is at least 3 in [61] and at least 28 in [62], which is far from the optimum and may not be practical in many applications.

4.3 Theoretical Results

In this section we consider a combinatorial structural property of replacement paths: such a path is a concatenation of a few original shortest paths. As we previously described, later on our deep learning algorithm builds on this lemma.

Our research is motivated by the following lemma from Afek *et al.*.

Lemma 1 *[63] After k edge failures in an unweighted graph, each new shortest path is the concatenation of at most $k + 1$ original shortest paths.*

In other words, the replacement path can be defined using so called pivot nodes that specify at which nodes in the graph the shortest paths may be stitched together. In this work we are interested in the failure of a single node, which is equivalent to the failure of its incident edges. The number of concatenations (and with that the number of corresponding pivots) required to obtain the replacement path then depends on the degree of the failed node. While in real-world networks the average degree is often rather small, finding suitable pivots remains a hard task. To overcome this problem, we consider an approximate setting, where we allow for a slack in the quality of the obtained paths (they may be longer than a shortest replacement path) but where only one pivot node is used. From the theoretical perspective, the following lemma is a special case of Lemma 1 for the case of a single edge failure.

Lemma 2 *After an edge failure in an unweighted undirected graph, each new shortest path is the concatenation of at most two original shortest paths.*

Given (s, t, f), let $P(s, t, f)$ be a shortest path from s to t in $G - \{f\}$. According to Lemma 2 it follows that $P(s, t, f)$ is a concatenation of two original shortest paths, or in other words, there exists a pivot vertex v such that $P(s, t, f)$ is the concatenation of the two shortest paths $P(s, v)$ and $P(v, t)$, where $P(s, v)$ is a shortest path from s to v in G and $P(v, t)$ is a shortest path from v to t in G. Motivated by this lemma, we will assume that it is sufficient to approximate the replacement path of a node failure using a single pivot node. As mentioned above, in the remainder of this chapter we show that it is possible to use deep learning to find such pivot nodes.

Network Name	**Nodes**	**Density**	**Average Degree**	**Dataset Size**
chem-ENZYMES-g118	95	2.71E-02	2.547	9.03E+03
chem-ENZYMES-g296	125	1.82E-02	2.256	1.56E+04
infect-dublin	410	3.30E-02	13.488	1.00E+05
bio-celegans	453	1.98E-02	8.940	1.00E+05
bn-mouse-kasthuri-graph-v4	987	3.16E-03	3.112	1.00E+05
can-1072	1,072	1.18E-02	12.608	1.00E+05
scc_retweet	1,150	9.98E-02	114.713	1.00E+05
power-bcspwr09	1,723	2.78E-03	4.779	1.00E+05
inf-openflights	2,905	3.71E-03	10.771	1.00E+05
inf-power	4,941	5.40E-04	2.669	1.00E+05
ca-Erdos992	4,991	5.97E-04	2.977	1.00E+05
power-bcspwr10	5,300	9.66E-04	5.121	1.00E+05
bio-grid-yeast	6,008	8.70E-03	52.245	1.00E+05
soc-gplus	23,613	1.41E-04	3.319	1.00E+05
ia-email-EU	32,430	1.03E-04	3.355	1.00E+05
ia-wiki-Talk	92,117	8.50E-05	7.833	1.00E+05
dbpedia-occupation	127,569	3.08E-05	3.934	1.00E+05
tech-RL-caida	190,914	3.33E-05	6.365	1.00E+05

Table 4.1: Statistics of real-world networks for DSO, sourced from the Network Repository.

4.4 Datasets

We test our method on a variety of undirected, unweighted networks from the Network Repository [48], which covers a diverse set of areas, such as road networks, biological networks, and communication networks, all representing potential input for real-world applications. For a list of all considered networks, we refer to Table 4.1.

The training schema of each comparison model was used, but the testing dataset was standardized for the purposes of a fair comparison. Specifically, $min(10^5, n^2) * 0.10$ node pairs (a, b) were randomly sampled without replacement from each network, such that $a \neq b$.

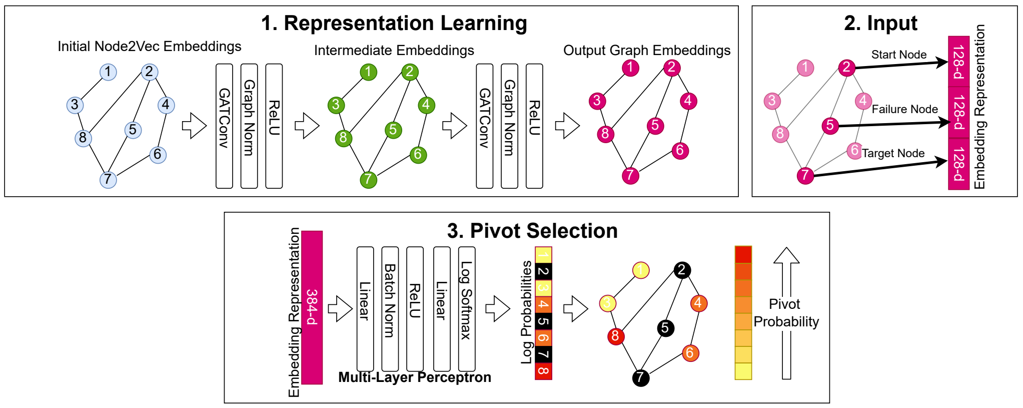


Figure 4.1: The DSO neural network architecture.

4.5 Method Summary

For learning a representation of the graph, we used a Graph Neural Net (GNN) model with a Graph Attention (GAT) layer [64]. The motivation behind this choice is that it is a modern embedding technique designed specifically with graphs in mind, with the specific graph structure guiding the connections in the neural network. GAT is a technique to refine the initial embeddings for the task of distance sensitivity oracles rather than the general purpose. While it has been used successfully for tasks like biological link prediction and text classification, to the best of our knowledge, we are the first to apply the GAT mechanism to shortest path problems. Similar to our approach in the previous project, we initialize these embeddings with Node2Vec as we found it advantageous for speed and accuracy performance.

Then, we entered a triplet of the embeddings of the source, target, and failed nodes into our oracle model, which consists of a two layer multi-layer perceptron with a softmax final layer to give a score to each node as a potential pivot. During back-propagation, the gradient is passed through the MLP and to the relevant embedding model parameters, so that the representations learned to encode task-specific features.

For training, we did pre-compute the true distances between points for each setting using a landmark system similar to the approach in the previous chapter.

Network Name	**MRE (Ndist2vec)**	**MRE (Rizi)**	**MRE (Random Pivots)**	**Representation Factor**	**MRE (Our Method)**
chem-ENZYMES-g118	100.00%	14.19%	215.96%	811.89	0.27%
chem-ENZYMES-g296	100.00%	7.85%	257.69%	530.22	0.49%
infect-dublin	38.33%	13.58%	191.02%	1,091.55	0.18%
bio-celegans	15.45%	9.84%	176.28%	4,299.51	0.04%
bn-mouse-kasthuri-graph-v4	17.15%	16.91%	204.20%	5,105.11	0.04%
can-1072	36.43%	11.23%	211.58%	573.40	0.37%
scc_retweet	100.00%	10.78%	167.64%	3,287.11	0.05%
power-bcspwr09	42.75%	20.45%	237.46%	1,493.46	0.16%
inf-openflights	100.00%	16.32%	202.05%	280.63	0.72%
inf-power	59.08%	31.67%	228.67%	1,013.10	0.23%
ca-Erdos992	100.00%	18.93%	206.26%	630.77	0.33%
power-bcspwr10	37.02%	31.95%	232.83%	739.13	0.32%
bio-grid-yeast	14.67%	15.95%	173.80%	27.80	6.25%
soc-gplus	100.00%	55.99%	200.20%	654.61	0.31%
ia-email-EU	13.31%	119.61%	202.93%	216.11	0.94%
ia-wiki-Talk	20.83%	28.19%	203.29%	26.50	7.67%
dbpedia-occupation	Did not finish	28.41%	205.07%	31.56	6.50%
tech-RL-caida		51.92%	206.02%	26.04	7.91%

Table 4.2: Results of models across several real-life networks.

We used a large variety of datasets provided by Network Repository [48] and compared our results to Rizi et. al. [39] and Chen et. al [47]. All embeddings generated were 128 dimensional.

4.6 Experimental Results

In line with previous works computing shortest paths with deep learning, we evaluate our method using the **Mean Relative Error (MRE)** metric. Let $\hat{d}_{i,j}$ denote the predicted distance and $d_{i,j}$ denote the actual distance. The Relative Error is then given by

$$RE = \frac{|\hat{d}_{i,j} - d_{i,j}|}{d_{i,j}}$$

We note that, for evaluations of DSOs, $\hat{d}_{i,j}$ and $d_{i,j}$ denote the predicted and actual distances on the graph after a node failure.

We also provide the representation factor, denoting the ratio of the MRE obtained with random pivots and the one obtained using our method, as a metric for evaluating the quality of our representations.

Across all networks, we were able to match or outperform the state-of-the-art works, often by several degrees of magnitude (Table 4.2). We did so with fewer training cases and no special selection process. In doing so, we demonstrated that deep learning can be used to effectively find the shortest replacement paths.

We also tested our method against random pivots, to combat the possibility that the graphs we were studying were simply very dense, with a high population of cliques where most nodes would be suitable pivots. Our model significantly out performs this case as well which supports our conclusions.

4.7 Conclusion

We have shown that distance sensitivity oracles with close to optimal performance can be obtained by utilizing the power of deep learning. Our method builds on a combinatorial property that allows for finding replacement paths based on pivot vertices. On a variety of real-world networks in the presence of failures, we can reliably find suitable pivots where the lengths of the corresponding replacement paths are very close to those of optimal paths. Moreover, our experiments suggest that these results are not artifacts of the inherent structure of the inputs, but are instead based on the fact that the different building blocks of our pipeline successfully capture the relevant structural information about the input graph.

As a consequence, it would be interesting to apply this method to related tasks where similar structural information needs to be captured. One such example is *local routing*, where the goal is to find short paths in a graph without the use of a central data structure by greedily routing to nearby embeddings. Prior work has shown that close to optimal greedy routing can be performed when embedding networks into hyperbolic space [65]. However, the resulting embeddings were susceptible to numerical inaccuracies, and network failures decreased routing performance a lot. It would thus be interesting to see whether our approach can be extended to the greedy routing setting as well, in order to overcome the previously observed issues.

Additionally, our approach has currently not been tested on larger networks containing millions of nodes. By calculating the APSP information using a distance oracle and using an improved node2vec implementation, we plan to test our networks' scalability in the future.

Chapter 5

Stochastic Block Model

5.1 Abstract

In this project we propose to expand upon a spectral algorithm for community detection in the stochastic block model first presented by Chin et. al. [66] by considering additional variations of construction. Our algorithm works with graphs that contain unequal sized blocks and non-uniform densities which was not considered by the original algorithm. It builds upon the spectral algorithm and remains robust and simple while capable of solving additional cases.

5.2 Background

The Stochastic Block Model (SBM) is a widely studied model for community detection in graphs. The graph is constructed with latent community structure where nodes exhibit higher likelihoods of connection within their respective blocks than between blocks.

More formally, in the classic stochastic block model, there are k blocks each of size n where edges are generated randomly based on the following distribution: Pairwise, there will be an edge between nodes (u, v) with probability $\frac{a}{n}$ if u and v belong to the same block, and with probability $\frac{b}{n}$ otherwise, where $a > b$. For simplicity we will focus on the $k = 2$

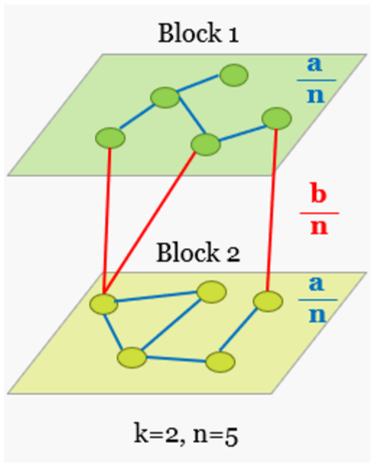


Figure 5.1: A visualization of the construction of the Stochastic Block Model

case to start.

If we denote the true blocks (sets of users within the same community) to be V_1 and V_2 in the $k = 2$ case, we want to find a partition of the vertex set V_1' and V_2' such that V_1 and V_1' are close to each other, and as are V_2 and V_2'.

Chin et. al. [66] showed that a simple spectral algorithm can find a partition with high probability under certain constraints on a and b, namely that they have to have a large enough gap to guarantee high fidelity community detection. This work was an improvement on previous work proved by Coja-Oghlan [67].

Theorem 3 *There are constants C_0 and C_1 such that the following holds. For any constantsa $> b > C_0$ and $\gamma > 0$ satisfying $\frac{(a-b)^2}{(a+b)} \geq C_1 log\frac{1}{\gamma}$, we can find a γ-correct partition with probability 1-o(1) using a simple spectral algorithm.*

Where the metric γ-correctness is a common tool for evaluating recovered partitions. It is defined as follows:

Definition 1 *Given a set of k true blocks V_i for $i \in [1, k]$, a reconstruction V_i' is γ-correct if $|V_i \cap V_i| \geq (1 - \gamma)n \; \forall i$*

Abbe [68] provided an extensive survey of recent results for SBMs and other block models, including an overview of algorithms for community detection, including spectral and probabilistic MAP techniques and precisely scope the limits of these approaches. Variations of the SBM are also studied in works such as [69] which considers community detection in a sparse hypergraph stochastic block model. [70] proposes a modeling system of dynamic social networks that is a temporal extension of the SBM.

In this chapter, we will discuss extensions of this result to variations of the simple spectral algorithm described by Chin et. al. [66] for alternative constructions of the SBM. We consider three main cases:

- Blocks of unequal size, where n is not uniform between blocks
- Blocks of unequal density, where a is not uniform between blocks
- A combination, where blocks have both different sizes and densities

These proposed variations represent a more realistic variation of the SBM with the goal of enhancing the community detection algorithm's performance on more general real-world networks.

5.3 Overview of the Spectral Algorithm for the Standard SBM

The spectral algorithm of [66] is motivated by the fact that the second eigenvector of the expectation of the adjacency matrix $\mathbb{E}(A_0)$ of the standard SBM separates the blocks. It is a rank 2 matrix with eigenvalues and eigenvectors

$$\lambda_1 = a + b, u_1(i) = \frac{1}{\sqrt{2n}} \forall i \in kn$$

$$\lambda_2 = a - b, u_2(i) = \begin{cases} \frac{1}{\sqrt{2n}} & \text{if } i \in V_1 \\ -\frac{1}{\sqrt{2n}} & \text{if } i \in V_2 \end{cases}$$

They observed that if they can find an approximation of the second eigenvector of $\mathbb{E}(A_0)$ then they can determine which nodes belong to which block based on its entry in the eigenvector. Therefore, the goal of the algorithm is to approximate this second eigenvector u_2 of the expectation of the adjacency matrix $\mathbb{E}(A_0)$. The spectral algorithm consists of first finding the vector space spanned by the top two eigenvectors of the adjacency matrix A_0. Since the first eigenvector of $\mathbb{E}(A_0)$ is the all ones vector, they take the projection of the all ones vector onto this space and approximate the second eigenvector by taking the vector that is orthogonal to this projection. Then, they sort the top k values and assign the corresponding nodes to V_1' and the bottom k to V_2'. Finally, there is a local correction step where nodes are reassigned to the block designated by the majority of their neighbors.

In this community detection problem, this local correction step is so powerful that typically any result that is close enough to the correct partitions can be vastly improved with this extra step. In their original paper [66] showed that this simple method achieves the result proposed in Theorem 3.

For the remainder of this chapter, we will consider extensions of this algorithm for variations where the basic assumptions of the SBM are broken, and we will explore how the spectral algorithm can be adapted to work in these cases.

5.4 Variation 1: Blocks of unequal size

The first variation we consider in this chapter is the case where the blocks are not the same size. In this modified SBM, the network is still partitioned into distinct blocks each representing a community. However, the sizes of these blocks can differ, reflecting the heterogeneous nature of real-world networks. We will require an additional parameter to define the network beyond the standard model, in this case the extra size variable. First we can consider the $k = 2$ case where the block sizes will be defined as n and m. The edge probabilities can no longer be defined as $\frac{a}{n}$ and $\frac{b}{n}$ as a result of the change, and we would like to keep the edge densities within communities consistent. Therefore, we introduce p as the probability of within-block edges and q as the probability of between block edges, where $p > q$. See Figure 5.2 for a visual representation of the expectation matrix in this variation.

Motivated by the previous result of [66] we will first consider the eigenvectors of $\mathbb{E}(A_0)$. Immediately, it becomes clear that the first eigenvector is no longer the all ones vector, and in fact, the first eigenvector separates the blocks of this matrix.

In this case, we will get an eigenvector of $\mathbb{E}(A_0)$ where values associated with vertices in V_1 will have value x and vertices in V_2 will have value y: $\begin{bmatrix} pmx + qny \\ qmx + pny \end{bmatrix} \sim \begin{bmatrix} x \\ y \end{bmatrix}$. Solving for $[x, y]$:

$$y(pmx + qny) = x(qmx + pny)$$

Then, without loss of generality we can set $y = 1$ and after some algebra:

$$qmx^2 + p(n - m)x - qn = 0$$

$$x = \frac{p(m - n) \pm \sqrt{p^2(n - m)^2 + 4q^2mn}}{2qm}$$

The eigenvector corresponding to the largest eigenvalue will be the addition case. Since all p, q, n, m are defined by the problem, this eigenvector can be explicitly computed. Note that it will never be 1, unless $n = m$ or $p = q$, both of which we assumed to not be the case.

Now, we don't need to find the vector space spanned by the top two eigenvectors of A_0, we can simply compute the top eigenvector of A_0 and the Davis Kahan sinΘ theorem from [71] [72] allows us to bound the angle between the first eigenvector of $\mathbb{E}(A_0)$ and the first eigenvector of A_0.

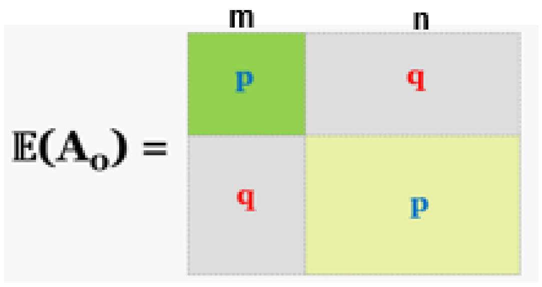


Figure 5.2: The expectation of the adjacency matrix in variation 1 where block sizes vary; one block is size m and the other is size n.

The new strategy thus becomes:

1. Remove all high degree rows and columns in the adjacency matrix
2. Compute the first eigenvector of the adjacency matrix
3. Take the top n entries of the eigenvector and assign its corresponding vertices to V_1' and the rest to V_2'
4. Apply local correction by polling the neighborhood of each vertex. If it has more neighbors in the other block, move it to that block.

To prove that this will work with high probability, we define error $E_0 = A_0 - \mathbb{E}(A_0)$. First we must delete rows/columns in A_0 that are outliers with degree $> 20(p+q)(n+m)$ because they disproportionately contribute to the operator norm, causing $||E_0||$ to be too large. Let $A, \mathbb{E}(A)$, and E be the matrices obtained from deleting high degree rows and columns from $A_0, \mathbb{E}(A_0)$, and E_0 respectively. Define $\Delta = \mathbb{E}(A) - \mathbb{E}(A_0)$; then we have:

$$A = \mathbb{E}(A) + E$$

$$A = \mathbb{E}(A_0) + \Delta + E$$

The goal now is to show that Δ and E have a small contribution to the operator norm. It is a simple process to bound Δ, because very few nodes will have high degree.

We use the following theorem, from [66] via a Chernoff bound:

Theorem 4 *There exit a constant d_0 such that if $d \geq d_0$ then with probability $1-exp(-\Omega((p+q)^{-2}(n+m)))$ not more than $(p+q)^{-3}(n+m)$ vertices have degree $\geq 20d$*

We take $d = (p+q)(n+m)$. From the lemma, there are at most $(p+q)^{-3}(n+m)$ vertices with degree $\geq 20(p+q)(n+m)$, which are precisely the vertices who's corresponding rows and columns in the original adjacency matrix will be recorded in Δ, by definition. Then, Δ has at most $2(p+q)^{-3}(n+m)^2$ non-zero entries, $2(n+m)$ for each vertex, because we zero out the entire row and column in A_0. The magnitude of each entry is at most $p+q$ so taking the norm of Δ, we get:

$$|\Delta|_{HS} \leq \sqrt{\frac{2(n+m)^2}{p+q}} = O(1)$$

This is sufficient because the Operator norm is bounded by the Hilbert-Schmidt norm.

Now, we will bound $|E|$ using the following lemma from [66] adapted from [73] and [74]:

Lemma 5 *Suppose M is a random symmetric matrix with zero on the diagonal whose entries above the diagonal are independent with the following distribution:*

$$M_{ij} = \begin{cases} 1 - p_{ij} & \textit{w.p. } p_{ij} \\ -p_{ij} & \textit{w.p. } 1 - p_{ij} \end{cases}$$

Let σ be a quantity such that $p_{ij} \leq \sigma^2$ and M_1 be the matrix obtained from M by zeroing out all the rows and columns having more than $20\sigma^2 n$ positive entries. Then with probability 1-o(1), $||M_1|| \leq C\sigma\sqrt{n}$ for some constant $C > 0$.

We can see that this structure is exactly the construction for the matrix E. It directly follows that $|E| \leq C\sqrt{(p+q)(n+m)}$ for some C with high probability.

Now we can apply the Davis-Kahan sinΘ Theorem to get:

$$sin(< u_1, u_1') \leq \frac{E + \Delta}{\lambda}$$

If we bound $\lambda = (p-q)(m-n)$ by $C_2\sqrt{(p+q)(n+m)}$

$$sin(< u_1, u_1') \leq \frac{C\sqrt{(p+q)(n+m)}}{C_2\sqrt{(p+q)(n+m)}} = c$$

Assuming this condition on λ, and thus that the gap between $pm + qn$ and $qm + pn$ is sufficiently large, we can see that the angle between the two vectors: the first eigenvector of $\mathbb{E}(A)$ and the first eigenvector of A is small. Therefore, we can use it to approximate the blocks.

The rest of the proof follows directly from the [66] paper, once this bound is achieved we can show a high probability of a γ-correct partition reconstruction.

5.5 Variation 2: Blocks of unequal edge density

For the second variation, we will focus on blocks of equal size that differ in edge density within blocks. Here, we will denote the size of the blocks as n, just like in the classic SBM setup. We will also return to the $\frac{a}{n}$ and $\frac{b}{n}$ notation for intra- and inter-edge probabilities respectively, but $\frac{a}{n}$ will be the edge probability for just vertices within V_1, and for V_2 we will use probability $\frac{c}{n}$. See Figure 5.3 for a figure of the expectation matrix for this variation.

In this case, we follow similar steps to compute the eigenvector of the expectation of the adjacency matrix $\mathbb{E}(A_0)$ as we see that:

$$\begin{bmatrix} ax + by \\ bx + cy \end{bmatrix} \sim \begin{bmatrix} x \\ y \end{bmatrix}$$

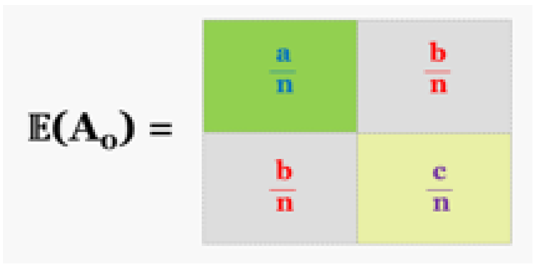


Figure 5.3: The expectation of the adjacency matrix in variation 2 where block sizes are the same but the probabilities of edges within blocks differ by block; one block has intra-block edge probability $\frac{a}{n}$ and the other has intra-block edge probability $\frac{c}{n}$.

$$y = 1$$

$$x = \frac{(a-c) \pm \sqrt{(a-c)^2 + 4b^2}}{2b}$$

Which we can easily verify would be 1 in the case that $a = c$. Again, we have that the all ones vector is no longer the first eigenvector, and indeed, the first eigenvector separates the blocks. We can follow a similar proof to the first variation to show that using the same alteration to the spectral algorithm will result in community detection with high probability of correctness.

5.6 Variation 3: Combined Case

In the combined case, we consider a scenario where the blocks have both unequal sizes and edge distributions. A visual representation of the expectation of the adjacency matrix in this situation is shown in Figure 5.4. Here, we define our two blocks to have sizes m and n respectively, similar to the first variation. Additionally, similar to the second variation, block one will have edge probability within blocks p and the second block will have probability r within the block. For any two vertices not within the same block, the edge probability will be denoted q.

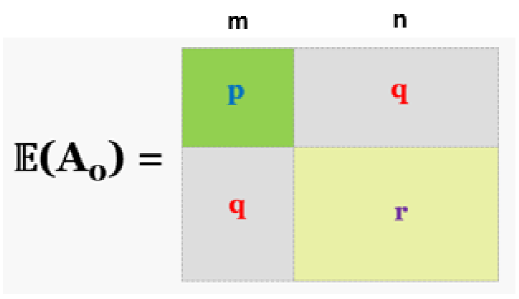


Figure 5.4: The expectation of the adjacency matrix in variation 3 where both block sizes and edge probabilities differ. One block is size m with intra-block edge probability p and the other is size n with intra-block edge probability r.

Following similar steps to the previous two cases, we can compute the eigenvectors of $\mathbb{E}(A)$ to observe that:

$$\begin{bmatrix} pmx + rny \\ qmx + pny \end{bmatrix} \sim \begin{bmatrix} x \\ y \end{bmatrix}$$

$$y = 1$$

$$x = \frac{(pm - rn) \pm \sqrt{(rn - pm)^2 + 4q^2mn}}{2qm}$$

Assuming that $x \neq 1$ then we can show perfect reconstruction using the first eigenvector appoximation:

Theorem 6 *If $mp+qn >> log(n+m)$ and $\sqrt{(mp+nq)log(m+n)} << \frac{|(mp+qn)-(mq+rn)|}{2}$ then the proposed algorithm can partition communities with high probability.*

We want to show that we classify nodes to communities incorrectly with low probability. Where X_i is an entry in the adjacency matrix in row i, i will be misaligned with probability:

$$Pr(\sum_i X_i \geq \frac{(mp+qn)+(mq+rn)}{2}) =$$

$$Pr(\sum_i X_i \geq (mp+qn) + \frac{-(mp+qn)+(mq+rn)}{2})$$

From the assumption in the theorem statement, this is bounded by the following:

$$<< Pr(\sum_i X_i \geq (mp+qn) + \sqrt{(mp+nq)log(m+n)})$$

Take $\lambda = mp + nq$ and $C = 100\sqrt{log(m+n)}$, for example

$$Pr(\sum_i X_i \geq \lambda + C\sqrt{\lambda})$$

Then, from the Chernoff Inequality stated in [75], we see that this is bounded:

$$Pr(\sum_i X_i \geq \lambda + C\sqrt{\lambda}) \leq exp(-\frac{C^2\lambda}{2(\lambda + C\sqrt{\lambda}/3})$$

Since $C = 100\sqrt{log(m+n)} < \lambda^2$ by our first assumption:

$$Pr(\sum_i X_i \geq \lambda + C\sqrt{\lambda}) \leq exp(\frac{-C^2}{4})$$

$$= exp(-100^2 log(n+m)/4) = (n+m)^{-2500}$$

This probability is very small compared to $n+m$, so it is very probable that perfect reconstruction can be achieved in the general case under the theorem's assumptions.

If we additionally employ the correction step, this should remove the log term in the bound via the union bound over all sets of vertices, similarly to the result of [66]. The intuition behind this is that C can be a much larger factor since more vertices are allowed to fail at the step prior to local correction to achieve close to perfect reconstruction guarantees.

However, this general case is more complicated than the previous two, because it is possible to observe a situation where $x = 1$ if the equality: $pm + qn = qm + rn$ holds. In this situation, we would indeed need to explore the second eigenvector algorithm proposed originally by [66].

Therefore, in this combined case we need to employ an adaptive strategy that is robust to this equality becoming close to true. We implemented the algorithm in software and found experimentally that as expected, when the equality is not satisfied, the blocks are easily reconstructed, as shown in Fig 5.5. Similarly, when the equality is satisfied, the blocks cannot be reconstructed with the first eigenvector approximation and we must use the second eigenvector instead, see Fig 5.6.

An interesting case arises when the equality is close to true, but not completely satisfied, as shown in the example in Fig 5.7. If the first eigenvector algorithm is able to find any partition, even if it is incomplete, we can apply iterated correction steps until convergence to further improve the community recovery.

Therefore, the adaptive strategy becomes:

1. Run the approximation algorithm of the first eigenvector.
2. If the algorithm predicts any two communities, no matter their sizes, run the correction step repeatedly until the communities converge.
3. If the algorithm predicts only one community, then we approximate the second eigenvector of the expectation of the adjacency matrix to separate the blocks.

This strategy is robust to general variations of unequal block sizes and densities within the Stochastic Block Model, filling in a gap from the original paper [66].

5.7 Conclusion

In this chapter, we have provided an overview of a simple spectral algorithm for community detection of the stochastic block model (SBM) and its variants. We have underscored the importance of considering realistic variations of the SBM, such as those accommodating

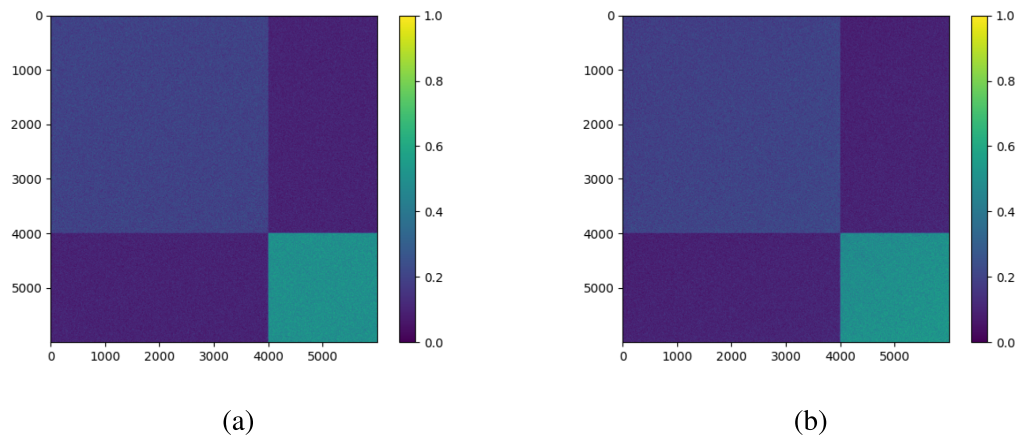


Figure 5.5: Variation 3 (unequal block size and density) example: a) Original Adjacency Matrix, b) Reconstructed Adjacency Matrix demonstrating a full reconstruction with the first eigenvector approximation method. This example was generated with m=4000, n=2000, q=.1, p=.2, r=.5

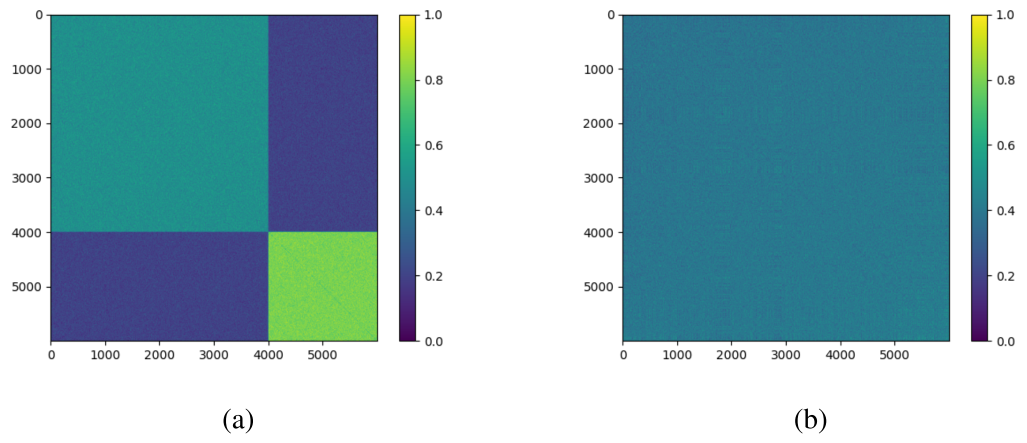
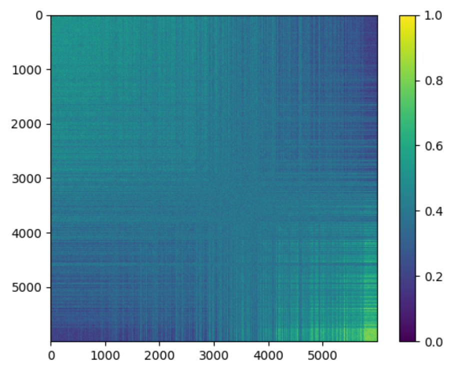


Figure 5.6: Variation 3 (unequal block size and density) example: a) Original Adjacency Matrix, b) Reconstructed Adjacency Matrix demonstrating a failed reconstruction with the first eigenvector approximation method as the first eigenvector of the expectation of A is the all ones vector. In this case, we will need to use the second eigenvector approximation to separate the blocks. This example was generated with m=4000, n=2000, q=.2, p=.5, r=.8


(a)

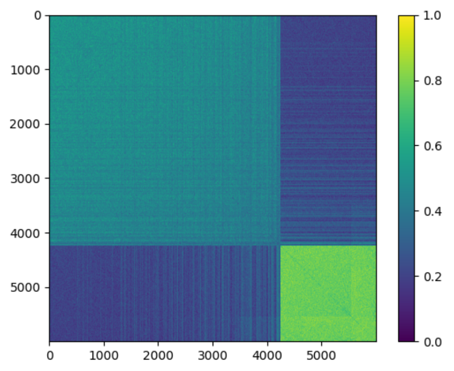

(b)

Figure 5.7: Variation 3 (unequal block size and density) example: a) Reconstruction after one Correction step, b) Reconstruction after two Corrections. In this case, we observe an intermediate stage where the first eigenvector approximation is able to identify a portion of the blocks but not completely successfully. With repeated corrections, the reconstruction improves. This example was generated with m=4000, n=2000, q=.2, p=.5, r=.775

unequal block sizes and edge densities. By embracing such natural extensions, we hope to make the classic spectral algorithm more versatile, and to better capture the nuanced structure of real-world networks which often exhibit heterogeneous community organization. This refinement is motivated by the enhancement of the fidelity of SBM-based models to empirical data and also enriches our understanding of the limits of this approach.

We focused on the $k = 2$ case specifically but the algorithms discussed here can easily be extended to the general case. We have not implemented the general case in software yet, but this is something that we would like to complete as a next step in this project.

Some potential future work involves considering additional cases, such as when we do not have complete information about the graph. In this project, we assumed that all of the parameters are known but this is not realistic for community detection tasks over real-world networks. We could consider a case like the third variant proposed here, but where the values of $p, q, r, m,$ and n are not known. The algorithm at its core is still sound under this scenario, but it does have repercussions for specific implementation and will likely affect experimental results.

Additionally, it would be illuminating to explore further practical applications of this method. We implemented and tested the algorithm for the combined case on synthetic graphs, but it would be a natural next step to empirically examine how well this method performs on real-world networks, as the algorithm extension presented here is more aligned than other previous iterations of methods for this problem.

Chapter 6

Cross-Network Alignment

6.1 Abstract

Following the problem of community detection, we can consider the problem of network alignment. Separate social networks may be aligned via their shared users, and many users are friends on multiple networks. Cross-Network Alignment attempts to find aliases of users between networks based on social network structure and potentially semantic information present in the dataset. In this section, we present three distinct approaches to detect cross-network associations of users with various limitations and perspectives.

6.2 Background

The formulation of the cross-network association problem is that there are multiple graphs that represent similar or related data, and the goal is to associate nodes between the graphs. A simple example of this is a graph of Facebook users and a graph of Twitter users where our goal is to discover a partial mapping between accounts on these social networks. This mapping would indicate that for a pair of users in the map, the Facebook account is an alias for the same user on the mapped Twitter account.

A user will generally have accounts on multiple different websites and engage in con-

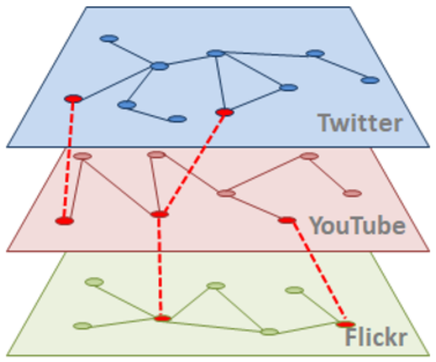


Figure 6.1: A visualization of the cross-network alignment problem, where red edges between networks representing the same entity are what we are hoping to uncover.

tent and form relationships in connected but not identical ways. This multi-platform presence creates a rich environment of interconnected digital footprints, offering a wealth of data for analysis, modeling, and integration. Cross-network alignment can provide a wider picture of a user and their network, and it has important applications to ad recommendations, modeling the spread of information across networks, and surveillance for security threats.

However, these social networks are inherently heterogeneous because users engage with different social networks for different purposes and in different contexts. Unlike a controlled setting with synthetic data where graphs are close to isomorphisms, not all users will be present in every social network. To make matters more complex, researcher's tools for gathering social network data also often result in incomplete data captures of the networks due to space limitations and potential privacy settings.

Despite the difficulty of the problem, it has been a topic of interest in graph theory over the last two decades and significant advances have been made. There are three common categories of approaches to the problem: 1) A spectral approach where the output of the algorithm is a score for each pair of nodes. Some examples include REGAL [76], FINAL [77], IsoRank [78], MAD [79] and BigAlign [80]. 2) Algorithmic or combinatorial approaches which are often greedy and rely on neighborhood similarity and semantic data such as username scores such as UIA [81], the work of Buccafurri et. al. [82], and FRUI [83]. 3) Graph embedding approaches which attempt to align the vector space of the graphs, typically with machine learning techniques. Examples of this technique include PALE [84], IONE [85], Deeplink [86], and COSNET [1].

A common limitation of some of these algorithms is that they rely heavily on semantic data, which may be unreliable if the user is attempting to conceal their identity or if semantic data is unavailable. Additionally, many of these aforementioned methods are supervised meaning they rely on ground truth data for analysis, which is not often available in practice.

I have researched this problem from three different angles, and will review each in detail in this chapter.

6.2.1 Problem Definition

Let a social network graph be denoted $G = (V, E)$ where V is the set of N nodes, or users, and $E \subseteq VxV$ is the set of within-network edges, or friendship links. For this project, we will assume that the graph is undirected meaning that $(u, v) \in E \rightarrow (v, u) \in E$. Each vertex $v_i \in V$ will have a representation in d-dimensions $R_i \in \mathbb{R}^d$, where R is a matrix containing all of the representations of V.

Here we consider two networks, a source and a target network which we will denote as $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ respectively. For each node in the source graph, we want to find its alias in the target graph assuming that it exists, which manifests as a mapping $M : u \in V_1 \rightarrow v \in V_2 \cup \{\emptyset\}$.

6.3 Datasets

Throughout this project, we used many cross-network datasets including PKDD12, KDD15, WDD13, and COSNET. Each dataset contains cross-network data with varying features. For example, WWW13 does not contain usernames but contains biographical data.

The main two datasets we considered are PKDD12 and COSNET because they contain cross-network truth data for training and evaluation of models.

6.3.1 PKDD12

Buccafurri et. al. [82] published PKDD12, a truth cross-network data set that contains friendship link information within five social networks: Twitter, Livejournal, YouTube, Flickr, and Google Plus. It also contains ground truth data of aliases between social networks, which they dub "me edges". The data set contains 90000 users between data sets and 745 me edges. We have primarily used this data set for preliminary testing as it is relatively small and contains cross-network truth data.

6.3.2 COSNET

COSNET [1] is a larger data set with 7,209,113 users over all of the networks. It contains friendship link information between five social networks: Flickr, Last.fm, Linkedin, Livejournal, and Myspace. It also contains the largest amount of ground truth data which link the same user between social networks.

The COSNET data set is very similar in format to PKDD12 but it is significantly larger. It contains the most nodes, and the nodes are the most connected of all of the data sets we have considered for this project. It however does not include biographical data.

6.3.3 KDD15

The KDD15 [87] dataset consists of six social networks including Flickr, Facebook, Google Plus, LinkedIn, Myspace, and Twitter, with a total of 750,545 users. It contains ground truth data for aliases as well as biographical information but does not include friendship links.

In this project, we infer friendship edges from biographical information. This process was challenging because the data was social network specific. We used a bag of words model to map how similar users were to each other. If they shared attributes like occupation or location we would draw a friendship edge. However, this lack of a true graph dataset to analyze made this dataset less appealing. The advantage of this dataset is that it contains the most varied sample of social networks.

6.3.4 WWW13

The WWW13 [88] dataset mainly is constructed of post data which is time-stamped, semantic data. This uniquely allows us to infer connections between users based on @-mentions and replies but no explicit links are provided. It also contains biographical data associated with the users profiles.

The social networks contained in WWW13 are Yelp, Twitter, and Flickr, with a total of 991,410 users.

6.4 Method 1: Pair-wise Classification with Graph and Semantic Features

6.4.1 Experimental Methods and Initial Results

The first approach we attempted views cross-network alignment from the perspective of a classification problem. For any pair of nodes from separate social networks, the model is trained to determine if they represent the same entity. We start with an incomplete seed set of known aliases, and compare their friends pair-wise, the motivation being that if you are friends in one network, you are likely to be friends in another.

We used basic classifiers in machine learning, including Multi-Layer Perceptron (MLP), Perceptron, Support Vector Machine (SVM), and Passive Aggressive Classifier. The goal with this choice was to try many different techniques and determine experimentally which performed the best. For the feature set, we used various semantic and topological features of the social network graphs, such as:

- **Q-grams**: Username string similarity score based on the counts of q-length substrings in the username.
- **S Score**: The recursive neighborhood string similarity measure of Buccafurri et. al. [82]. It measures the username similarity of not just the nodes being compared but their neighbors, and so forth recursively with diminishing contributions to the metric.
- **Pagerank**: A classic technique for ranking nodes in graphs based on the number and quality of their neighbors, originally used to rank websites for Internet search engines.
- **Degree**: The number of neighbors of a given node.
- **Degree Centrality**: The normalized degree.
- **Betweenness Centrality**: The number of times a node is used in the shortest path between other nodes.
- **Closeness Centrality**: The sum of the lengths of shortest paths to each other node.
- **Bakry-Emery Curvature**: Two-hop network topology feature that measures the connectedness of neighbors.

The first two features represent semantic features, the username similarity of the pair in question as well as the username similarity of their neighborhoods. The remainder of the included features represented topological based metrics to detect similarity of network role and structure.

We were able to show fairly good performance with this method on three datsets, as shown in Figure 6.1, beating the baseline in accuracy for many cases of classifier choice.

The WWW13 dataset was the hardest for our algorithm to classify because it does not contain friendship links, so we inferred friendship links from post data (@-mentions on

Classifier	Metric	PKDD-12	KDD-15	WWW-13
SVM	Accuracy	.961	.964	.833
	F1	.940	.943	.637
MLP	Accuracy	.935	.996	.828
	F1	.906	.994	.625
Perceptron	Accuracy	.783	.940	.830
	F1	.566	.881	.716
Passive-Aggressive	Accuracy	.964	.964	.829
	F1	.948	.937	.712
DetectMe	Accuracy	.829		
	F1	.702		

Table 6.1: Performance of our first method cross-network alignment algorithm over multiple datasets and classifiers.

Twitter, shared place of review on Yelp, and shared location on Flickr). The classifier is very accurate for false aliases, but there are ample false negatives. This is likely due to the fact that we are inferring friendship links, so topology features are less reliable.

Similarly, KDD15 does not contain friendship links, but it also does not contain spatio-temporal post data like WWW13, making inferring links very difficult. We inferred friendship links from shared biographical information, for example users who were from the same location. The classifiers performed very well on this dataset despite the inferred links. We imagine this is because the usernames of the users in these networks (Facebook, Linkedin, etc.) are often tied to the users' real name rather than an avatar or online anonymous persona. Thus, the algorithm is able to classify mostly based on username similarity score.

For PKDD12, the Passive-Aggressive showed the best performance followed by SVM, MLP, and Perceptron. We were reliable able to achieve f-score of over .9 for this dataset using our features set. The DetectMe Algorithm of Buccafurri et.al. (which uses only semantic features) had an f-score of .7 on this dataset. This implies that the network topological features improve cross-network entity classification.

6.4.2 Non-Semantic vs Semantic Features Performance

However, we were curious whether the primary reason for the performance metrics we were observing was the username similarity. To gain insight into how heavily the model was relying on the semantic features (Q-grams and neighborhood similarity score), we eliminated those features and tried to classify node pairs with only network topology features: degree, degree centrality, betweenness centrality, closeness centrality, pagerank, and curvature.

There are still very few false positives, but it misses a majority of the true me edges that it is able to find using semantic features. With semantic features on the PKDD12 dataset, we observed an f-score of .938 but without semantic features on the same train/test set we observe an f-score of .377. This result implies that the topological features may have some measure of importance alone, but the username similarity scores are likely the most identifying feature.

We also experimented with removing the topological features from our feature set, using only Q-grams and S-score to determine gains from the topological features. The f-score is .664 and the accuracy is .824 with this setup on PKDD12. This result is not bad, but it is a decrease from the result with the topological features. This indicates that the topological features are adding information, but semantic features are very salient for this dataset.

This conclusion agrees with what we observed when we removed the semantic features.

6.4.3 Feature Sensitivity Analysis

In order to test which features the classifier was most sensitive to, we performed sensitivity analysis by removing each feature and observing the effect of the removal on SVM performance. As expected, the SVM performance was most sensitive to features that incorporate semantic information, the S-score and Q-grams.

Sensitivity Ranking	Feature
1	S-score
2	Q-grams
3	Betweenness Centrality
4	Degree Centrality
5	Bakry-Emery Curvature
6	Closeness Centrality
7	PageRank
8	Degree

Table 6.2: Sensitivity Ranking of Features. The recursive neighborhood username similarity measurement S-score, was the most significant feature for the SVM.

6.4.4 Summary and Conclusion

In this section, our semantic-based approach for cross-network association has demonstrated remarkable performance across a range of datasets. However, our feature sensitivity and non-semantic feature experiments suggest that much of this success can be attributed to the similarity scores derived from username strings rather than from graph topological features. In our subsequent method, we sought to address this challenge by excluding all semantic features, including usernames, from our analysis.

6.5 Method 2: Pair-wise Classification with Link Prediction and no Semantic Features

6.5.1 Method Summary

In our second approach to this problem, we removed the semantic features completely so that the model would have to rely solely on the topology of the network. Motivated by the previous projects discussed in this thesis, we employed a deep learning model with node embedding representations.

The approach uses two main steps: We learn node embeddings for each network with a single network link-prediction by performing random walks on both networks and learn node embeddings for G_1 and G_2 independently. We chose the link prediction model because it was a new state-of-the-art model at the time of the project that had a large focus on edge based learning tasks, which we determined to be promising for the cross-network alignment problem. Then we topologically align those embedding spaces by feeding pairs of learned embeddings from G_1 and G_2 into an deep neural network alignment model to get a score of how likely those entities are to be the same person. The alignment model is a two layer Multi-Layer Perceptron neural network, motivated by its strong performance in the first approach as well as other previous projects in this thesis as an accessible start for this problem.

Network	F-Measure	AUC
Flickr	0.78	0.90
Last.fm	0.86	0.94
Myspace	0.77	0.83
LinkedIn	0.86	0.95

Table 6.3: The result of our link prediction model on the COSNET dataset

The link prediction technique utilized in this study draws its foundation from Word2Vec [89], where embedding vectors are learned for each word in a corpus. By setting a window size within which words frequently appearing in similar contexts are positioned closely in embedding space, Word2Vec facilitates semantic similarity representation. Node2Vec [20] applies the concepts from Word2Vec on graph structured data, but instead of a window of words, a random walk in the graph is used as the context. This process allows relational similarities between nodes traversed in these walks to be captured and consequently, nodes that frequently occur in the same random walk are close in the embedding space. We use a Multi-Layer Perceptron (MLP) model with the embeddings as input to predict future edges for link prediction. The advantage of this approach is that it provides an edge-centric view of the graph via link prediction which will be valuable for cross-network association.

For this project, we used the method developed by Google Research [90] which learns node embeddings and MLP weights simultaneously. This approach demonstrated superior performance over various conventional link prediction methods, particularly with smaller embedding dimensions.

To apply the Google Research [90] method to the COSNET data, we create a test set by removing edges randomly while maintaining connectivity in the graph. After the edges in the test set are removed and stored, the remaining graph is the training set. To create negative edges for training, random edges are proposed and checked to ensure they were not already in the edge set. For training we made 50% of the edges positive and 50% negative examples. Random walks are then conducted on the training graph, with the resulting train/test sets and random walks serving as inputs to the MLP.

This embedding method achieves high link prediction accuracy for all COSNET networks, as seen in Table 6.3.

For the network alignment model, we used a Multi-Layer Perceptron as the classifier, similar to the approach in the first method. However, for this method we do not use the explicitly computed features at all. Instead, we input the concatenated pairs of embeddings from link prediction into the model. It is still a classification setting, so the model outputs a 1 or 0 depending on whether or not it believes the two nodes represent the same entity.

6.5.2 Experimental Results

The topological alignment embedding method yields solid cross-network accuracy and comparable results to the SVM baseline from our first method, all without semantic information, as shown in Table 6.4.

While the performance of our second method is notably inferior to Method 1 when considering the complete dataset with semantic features, a comparison with the results obtained using non-semantic features in Method 1 reveals a significant improvement with our link prediction-based embedding technique. This demonstrates the power of node representations and embeddings. The explicit topological features were much more computationally

Network Pair	F-Measure	AUC
LinkedIn - Flickr	0.76	0.74
LinkedIn - Last.fm	0.72	0.74
LinkedIn - Myspace	0.67	0.58
Flickr - Myspace	0.68	0.65
Flickr - Last.fm	0.72	0.80
Last.fm - Myspace	0.67	0.63

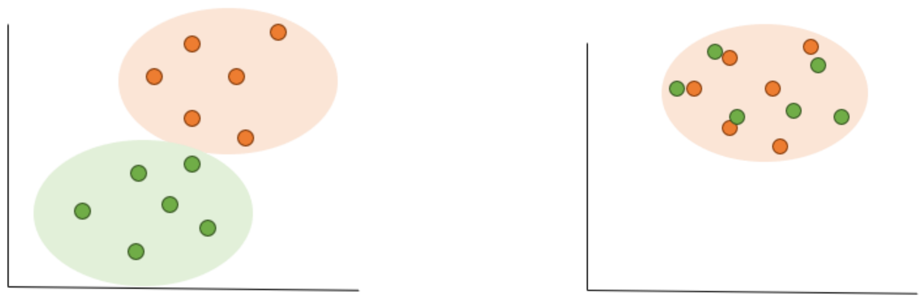
Table 6.4: Results of our topological alignment algorithm on the COSNET data set [1]

expensive to run and performed worse overall than the embedding techniques. Its original performance is mostly so strong due to the username and semantic data information.

6.5.3 Summary and Conclusion

In this section we described our second approach for cross-network alignment. We continued to view the problem as a classification problem with an MLP as the alignment model. However, we used node representations as features based on a link prediction model from Google Research that does not consider any semantic data at all. The results were extremely strong considering the lack of semantic data, which was by far the most salient feature in method 1. However, it does indeed provide worse performance overall.

Although the first two methods discussed in this chapter show promising results, they rely heavily on the use of training data and known aliases to start and are very sensitive to the quality of that initial data. However, we cannot expect to have ground truth data for the cross-network alignment problem in general. For the next step of this project we consider the problem from an unsupervised lens, with no semantic data and no ground truth. In this view, we are truly looking at a network alignment solution based solely on the connections present in the underlying graph structure.



(a) (b)

Figure 6.2: a) This graphic represents the embedding spaces of the two networks before mapping, and we can see that they are incomparable based on standard distance metrics and thus difficult to align. b) The result post-mapping the green network's embeddings to the orange network's embedding space. Now we can directly compare them for cross-network entity resolution.

6.6 Method 3: Unsupervised Network Alignment

6.6.1 Method Summary

In our third and final method, we use the link prediction model to generate embeddings similarly to method 2, but for the alignment model, we used a Generative Adversarial Network (GAN) [91] to create an initial mapping between the two network's embedding spaces. A GAN is a type of deep learning framework comprising two neural networks, a generator and a discriminator, engaged in a competitive process. The generator creates synthetic data samples, in this case mapped embeddings to the other network's vector space, while the discriminator evaluates whether these embeddings are really from that network or not. Through iterative training, the generator learns to produce an increasingly realistic mapping, while the discriminator becomes more adept at distinguishing between real and mapped embeddings. This adversarial process theoretically drives the model to generate a high-quality mapping between the two spaces in an unsupervised setting, as shown in Figure 6.2.

The third method summary is: 1) We learn node embeddings for each network with a single network link-prediction by performing random walks on both networks to learn node embeddings for G_1 and G_2 independently. 2) Then we topologically align those embedding spaces using a Generative Adversarial Network (GAN) to approximate a mapping M. 3) Finally, we select anchor pairs based the cosine similarity of the mapped embeddings from M to the source graph embeddings.

Cross-Network Topological Alignment

After computing embeddings enriched by the link prediction technique, we then attempt to align the networks using only the learning embeddings based on structural features of the graph as apposed to semantic features such as username and biographical data.

For this step, we utilize a Generative Adversarial Network (GAN) [91] as the model to align the network's embedding spaces. The GAN consists of models: a generator and a discriminator. The generator takes as input embeddings from the source graph G_1 and attempts to output corresponding embeddings in the embedding space of G_2. The generator uses the following loss function, that it learns to minimize during the course of training:

$$L = -\frac{1}{N}\sum_{i=1}^{N} log(1 - D(G(r_1^i))$$

Where D is the output of the discriminator and G is the output of the generator.

Concurrently, the discriminator evaluates the authenticity of these mapped embeddings compared to true embeddings from G_2. Through iterated training, the generator refines its output to produce a more realistic mapping and the discriminator becomes increasingly adept at discerning real and generated embeddings. It is trained with the following loss function that it wants to minimize:

$$L = -\frac{1}{N}\sum_{i=1}^{N} log(D(r_2^i) + log(1 - D(G(r_1^i))$$

The first term refers to the probability of incorrectly classifying real embeddings as fake and the second is the case where generated embeddings are classified as genuine.

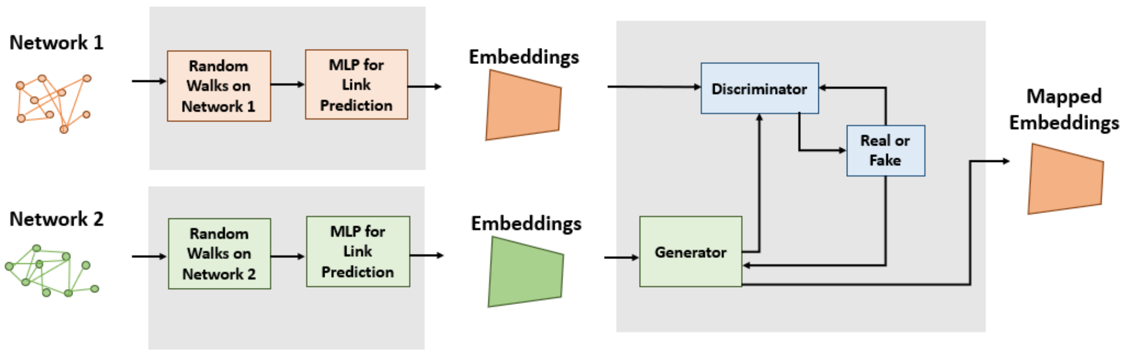


Figure 6.3: The architecture of the Link Prediction and GAN system for cross-network association.

Details of the architecture of the model can be seen in Figure 6.3. This adversarial learning format facilitates mutual improvement.

Association and Alias Prediction

Given a node $v_1 \in G_1$ and its mapped embedding $M(r_1)$ our goal is to find a corresponding $v_2 \in G_2$ that could represent its alias. To achieve this, we identify the embedding in R_2 that is closest to it. We tested with the Frobenius norm:

$$min_{r_2}||M(r_1) - r_2||, r_2 \in R_2$$

as well as the cosine similarity:

$$min_{r_2}cos(M(r_1), r_2), r_2 \in R_2$$

We then take the top k scoring pairs of nodes in the network based on an experimentally determined threshold to assign aliases. Alternatively, using these same metrics rather than simply taking the top score, we can generate a list of the top scoring aliases for each node to narrow exploration depending on the application.

6.6.2 Experimental Results

Metrics and Baselines

To evaluate our model for cross-network alignment we compared it to baseline methods on the COSNET dataset for Flickr and Last.fm. We used the metric of Precision@N based on the aliases discovered by the algorithm which is defined as follows.

$$P@N = \frac{\sum_{i=1}^{q} (\text{success@N(i)})}{q}$$

where success@N(i) is a binary variable representing whether or not the true alias of user i is within the top-N scoring candidates and q is the number of pairs for which we have ground truth evaluation data available.

We chose unsupervised methods of comparison including:

- **Degree-Based Alignment**: A trivial baseline where users in networks are aligned by relative degree rank.
- **Embedding Distance Alignment**: A baseline computed by applying closest distance matching to align the networks using the embeddings after link prediction without the GAN model that maps the embedding spaces. We used the cosine similarity for evaluation because it consistently performed better experimentally for the mapped embeddings.
- **MAD** [79]: A spectral approach to cross-network alignment that matches nodes via singular value decomposition, another comparable unsupervised model to predict aliases.

We also tested two different distance measures for the mapped embeddings of G_1 to G_2 embeddings as discussed in the previous section:

- **Frobenius Norm**

- **Cosine Similarity**

Finally, we tested the last.fm vs flickr alignment as well as sampled last.fm vs sampled last.fm with 80% common edges to test alignment in a less challenging setting.

Evaluation

In our first experiment, we aimed to assess the cross-network association by comparing random sampled versions of the Last.fm network. This setup ensures ample overlap in the graphs, in this case 80%, resulting in a test set that maintains equal core structure and similar characteristics while introducing variability and disrupting the isomorphism. By conducting the analysis on these sampled versions, we sought to evaluate the performance of our method in a more structured environment. For this experiment, we used Precision@1, Precision@5, and Precision@10. We chose such low values for k due to the heavy overlap resulting in the task being easier for the model.

The results of this experiment can be found in Table 6.5. It is evident that our method, Link Prediction + GAN outperforms both baseline methods- embedding and degree rank- for all tested cases. The substantial gap in performance in particular between the embedding and our full method demonstrates the power of the GAN model in aligning the two embedding spaces. We observe that the cosine similarity metric and the frobenius norm have very similar comparable results. This is expected, due to the fact that they operate on the same mapped embeddings. Consequently, the metrics are making the same decisions in most cases, leading to closely aligned scores. We do see that cosine similarity consistently slightly edges out the frobenius norm, which is an interesting result, indicating that the cosine similarity better captures the similarity of the embeddings. The MAD approach does beat our method for Precision@10 but their results are much worse for Precision@1 and comparable for Precision@5.

The results of our next experiment are summarized in Table 6.6. These results were generated from the alignment of the Last.fm and Flickr data with Flickr being mapped to

Model	Metric	P@1	P@5	P@10
Embedding	Cosine Similarity	0.0092	0.0379	0.1242
Degree		0.0081	0.0341	0.1039
MAD		0.1508	0.4040	**0.6905**
LP + GAN	Frobenius Norm	0.3009	0.4733	0.5704
LP + GAN	Cosine Similarity	**0.3056**	**0.4884**	0.6005

Table 6.5: Performance comparison between baseline methods for predicting aliases between sampled **Last.fm** and **Last.fm** networks with 80% overlap.

the Last.fm embedding vector space, meaning Flickr is the source network and Last.fm is the target network. In this experiment, we considered Precision@10, 20, and 30 to accommodate the more difficult task. Across all metrics, our method surpassed all other tested approaches in the cross-network alignment task, outperforming degree rank by the largest margin. We do observe for the first time, the frobenius norm performing better than the cosine similarity for one particular case.

Model	Metric	P@10	P@20	P@30
Embedding	Cosine Similarity	0.0506	0.1021	0.1262
Degree		0.0339	0.0975	0.1128
MAD		0.1341	0.1962	0.2870
LP + GAN	Frobenius Norm	0.1734	**0.2742**	0.3162
LP + GAN	Cosine Similarity	**0.1767**	0.2499	**0.3214**

Table 6.6: Performance comparison between baseline methods for predicting aliases between **Last.fm** and **Flickr** social networks.

Overall, these strong performances compared to baseline demonstrate the value in the LP + GAN approach proposed in this chapter. Given that the method is wholly unsupervised, the results are especially strong. We suspect that supervised methods would likely result in higher performance, but labeled training data is seldom available in real world networks. Additionally, the best pairs found by this unsupervised method could be used as a seed set or training set for other supervised methods.

6.6.3 Summary and Conclusion

In the third method, we transitioned from a classification-oriented perspective to a network alignment view. Here, our objective was to establish an unsupervised mapping between the original embedding spaces of two networks, aiming to align their vector spaces for anchor link identification.

Although there remains ample room for enhancement, this third method surpasses objective unsupervised baselines. While its performance falls short of analogous supervised models, it achieves this without leveraging any information beyond the network structure, thereby offering an exceptionally adaptable model.

6.7 Future Work

Potential future work could investigate the usage of Convolutional Graph Neural Networks as an alternative to explicitly computing embeddings. To our knowledge this approach has never been used before in the context of cross-network alignment, and it is theoretically an appropriate tool for this application as Graph Neural Networks implicitly extract high-level representations of graphs. Network embedding aims at representing nodes as vectors that preserve the topological structure of the original graph in order to allow for standard machine learning tools including classifiers like the multi-layer perceptron or an alignment model to be applied. Meanwhile, Graph Neural Networks are capable of performing network embedding problem through a graph autoencoder framework without the use of common techniques such as matrix factorization and random walks. As we have already completed a proof of concept of graph embeddings, this new method could potentially be applied to this problem.

There are many potential avenues for further growth of the method presented in this chapter. First, the aliases found by this method are not necessarily internally consistent, meaning that multiple nodes in G_1 can be mapped to the same node in G_2 and vice versa if we only consider top scoring aliases. We could implement a measure for global consistency derived from considering the set of distances in a systematic, non-greedy fashion.

In order to improve accuracy we could also experiment with additional embedding techniques including DeepWalk [92] and GraphSage [93]. We could alternatively attempt to link the embedding model to the alignment model and implement an iterative approach where we refine the original embeddings based on the mapping found by the GAN. The embeddings could be trained in tandem with the GAN to represent the graphs for the specific task of cross-network alignment.

Additionally, we are interested in investigating extensions and applications of this research. For example, the application to community detection within cross-network datasets. In each graph, only a subset of users in the entire multi-network are represented. More broadly, we can imagine identifying specific communities. Some may be friends on slack, some may connect on LinkedIn, but not everyone in the community is present on both networks. In this case, the community is most completely defined as the union of the sets within multiple social networks.

We could also extend this research for multiple networks, not just pairs. Currently to accommodate additional networks, a mapping would have to be trained pair-wise for every combination of two networks in the set. We would then have to assert global consistency in the results, making this scheme much more complex but very intriguing and potentially powerful.

6.8 Conclusion

This result has broad impacts and important consequences due to its practical application to real-world data sets such as social networks and its adaptability. The graph-based data model of social networks is commonly used in modern computer science research. This model is useful because it allows peopleâs profiles, activities, and information to be directly related with links, represented by edges. In these virtual societies, relationship links (such as friendship, following, commenting/reposting, or any other valid form of engagement) are the main form of expression individuals use to participate in the community. Studying social networks can allow us to track events, activities, information flow, and values within a community. However, social networks in the modern age are becoming increasingly complex. They are neither isolated nor independent; a user will generally have accounts on multiple different websites and engage in content and form relationships in connected but not identical ways. These various social networks are not the same, each may have different main functions or provide a space for niche content and we need to consider the entire story to perform in-depth analysis. An obvious example is if we are tracking a target of interest and we know their account on one social network, we could use this technology to discover their corresponding accounts on other social networks, allowing for increased surveillance of activity. Alternatively, if our goal is to track the spread of some information of interest, considering multiple networks is necessary because the information is going to be spread among different social networks potentially by the same users their followers. In general, we can use this association for increased accuracy and performance in our models. The implications of this research have a strong impact especially in the field of defense and security. In addition to being used directly in analyses for downstream tasks, the discovered aliases from the method proposed in this chapter could be used as seed sets for alternative cross-network alignment algorithms that take advantage of additional features but require ground truth for training.

Chapter 7

Geometric Wavelet Embeddings

7.1 Abstract

This section employs Geometric Multi-Resolution Analysis (GMRA) as a technique for dimensionality reduction and explores its impact on high-dimensional graphical learning tasks. GMRA exploits redundant representations in such high-dimensional datasets, embedding the high-dimensional data into an intrinsic, underlying lower-dimensional structure. This process aims to preserve essential features while reducing dimensionality and facilitate analysis by mitigating the computational complexities associated with analyzing high-dimensional datasets. This project proposes a novel application of Geometric Multi-Resolution Analysis to dimensionality reduction specifically in graph embeddings. Empirically, its efficacy if validated by its performance computing the intrinsic, underlying lower-dimensional structure for a comprehensive set of graph learning tasks, including node classification and anomaly detection in cyber network datsets.

7.2 Background

Throughout this thesis, we have discussed the application of graph embeddings for machine learning tasks on graphs in almost every project. This subsequent project aims to apply dimension reduction on graph embeddings to improve performance for general machine learning tasks over graphs. Dimension reduction techniques are extremely important in machine learning in general, because they have the potential to limit redundancy and improve model speed and size. Traditional methods for dimension reduction like singular value decomposition (SVD) and principal components analysis (PCA) are well-suited for data with linear structure. However, if the manifold is non-linear, more complex methods are required. For this project, we focus on an existing dimensionality reduction technique called Geometric Multi-Scale Resolution Analysis (GMRA) that has been shown to be advantageous on a wide variety of datasets[2] [94] [95].

GMRA was developed originally for reducing point clouds to projections onto many tangent planes, depending on scale, of the original space. Recent research has applied GMRA to alternative datasets and applications, such as classification of MNIST and CIFAR [96] where the images are flattened and interpreted as point clouds. It was also used on audio data via the MusicNet dataset [2] and on remote sensing images [97].

GMRA has never been used for graph embedding dimension reduction, which is the primary target of this novel project. High dimensionality in graph representations is a common problem for graph algorithms, where bloated representations can cause high complexity and increased computation time. As a researcher, it can be difficult to determine the optimal embedding dimension for datasets, so dimensions are chosen arbitrarily and tend to be quite high; for example 128 or 256 dimensional graph embeddings are very common in literature. Due to the promising results of GMRA in other contexts beyond point clouds, we were motivated to test the method for graph data and graph tasks.

We are working on this project under the DARPA CASTLE program, with the intention to integrate it with cyber security applications. Therefore, a the datasets used for analysis are related to the realm of computer network host level data.

Empirically, with GMRA we compute the underlying lower-dimensional structure of a selection of datasets and evaluate its consequential impact on a comprehensive gamut of graph learning tasks, including:

- Node Classification
- Edge Classification
- Link Prediction
- Anomaly Detection
- Graph Clustering

This was a joint work with Felicia Schenkelberg, who discussed this project in greater detail in her Dartmouth Master's Thesis. In this chapter, I will only discuss the subset of these collective findings that I directly analyzed, which is the Node Classification and Anomaly Detection results on cyber data. For more information and results of applications of GMRA, please consult Felicia's masters thesis.

7.3 Overview of GMRA

To implement GMRA dimension reduction, first, all of the points (in this case node embeddings) are inserted into a covertree data structure where each node is a point and an edge is drawn between two nodes if they are within euclidean distance 2^s of each other where s is the scale at each level. Constructing the covertree is the most computationally expensive part of the GMRA process. We will discuss an optimization for cover tree processing in the Batch Processing section.

Then, a Dyadic tree is created using the covertree as input. The dyadic tree clusters points by greedily picking all points within a radius, with the radius shrinking at lower levels of the tree to create levels of local geometry.

Finally, for each cluster in the dyadic cells, we perform SVD to compute the projection and wavelets for each node into a wavelet tree. Then, we can traverse the wavelet tree to extract embeddings at different scales.

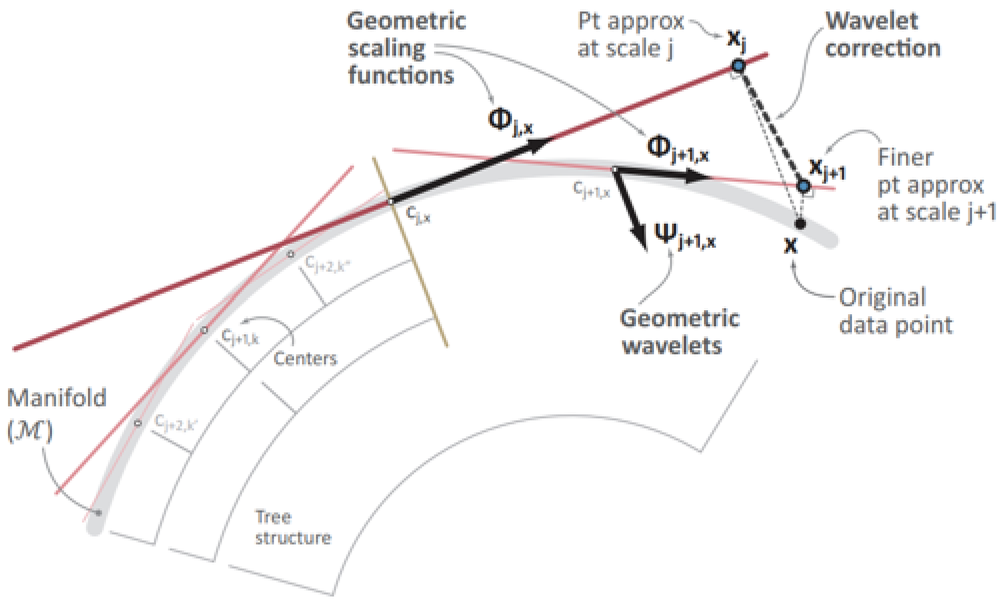


Figure 7.1: Image taken from [2], representing the GMRA wavelets at multiple scales that represent the original 2D points in 1D.

The result is a wavelet tree structure that we can traverse and that contains all of the data required to compute embeddings at reduced dimensions. The farther down the tree you go, the more wavelets (or tangent plane), and each wavelet will consist of few points. Each point in the original cloud is represented within exactly one wavelet node at each level. However, there is no guarantee that the wavelets will all be the same dimension ,it is determined by the intrinisc dimension computed by SVD for the set of points contained in that wavelet node.

7.4 Datasets

7.4.1 LANL

The LANL 2015 Comprehensive Multi-Source Cyber Security Events dataset [13] spans 58 days of log files of Windows-based authentication events within the Los Alamos National Laboratory's internal cyber network. It includes both normal activity and a redteam malicious campaign. We sampled 10 million edges over 10,000 nodes within a two day window of computer network traffic data. The full dataset contains 1,648,275,307 events in total for 12,425 users, 17,684 computers, and 62,974 processes. Some of the nodes are known to be malicious, which we used as truth data for node classification and anomaly detection of malicious behavior.

The format of the dataset is essentially an edgelist of a graph, where each node represents a process, desktop computer, server, or active directory server. Each event is on a separate line in the form of "time, source user@domain, destination user@domain, source computer, destination computer, authentication type,logon type, authentication orientation, success/failure".

We constructed a graph based on this data by drawing an edge between two nodes that had a communication event between them based on our sampled version of the dataset. The result is a directed Multi-Graph with 10 million edges and 10 thousand nodes.

7.5 Method Summary

In this section we will describe our experimental method to demonstrate the impact of GMRA embeddings on the performance of various graph tasks compared to the original, unreduced node embeddings.

The experimental procedure involved several key steps. First, to make the problem compatible with GMRA, we extracted node embeddings from the induced cyber graph. Then, we conducted baseline experiments with these embeddings using off-the-shelf models for cyber tasks which will allow us to establish a control for comparison to the GMRA embeddings.

Then, we apply the GMRA algorithm to the node embeddings which we interpret as a point cloud. Once we have the wavelet tree, we choose what we determine to be the best depth in the tree to extract embeddings. Then, we extract embeddings at that level for all nodes. The dimension at each level in the wavelet tree is dependent on the intrinsic dimension of the data and is not a controllable factor.

Once we have extracted the embeddings, we then run the same baseline experiments using the reduced dimension embeddings. The rationale behind this experimental method is to assess how GMRA influences the performance metrics for cyber tasks.

7.5.1 Initial Graph Embeddings

For the initial graph embeddings in this project, we experimented with GraphSage [93] and Node2Vec [20] embeddings to generate the point clouds representing our input graphs. Node2Vec is an embeddings technique based on random walks that we have used extensively thoughout this thesis. GraphSage is a more modern embedding technique that is essentially a Graph Neural Network (GNN), it aggregates neighbor features into an embedding. We used event counts as the node feature vectors for preliminary experiments.

The GraphSage embeddings displayed a significant dimension reduction compared to Node2Vec due to the large amount of redundancy in the embeddings, based on the small feature space they were constructed with. There were also many duplicate embeddings, which indicates that some network structure was lost during the embedding. For that reason, the majority of the experimental results are done with the Node2Vec embeddings as seeds.

7.5.2 Dimension Reduction and Reduced Embedding Extraction

We use the embeddings generated from Node2Vec as the input to the GMRA process, with the result being the wavelet tree structure. The process of extracting embeddings from a wavelet tree data structure is systematic. Each node in the wavelet tree, represents the description of a geometric wavelet, including the basis, scaling factor, node idxs clustered in that wavelet, and each node's wavelet coefficients for constructing their embeddings. We then navigate the tree to discover a location where all wavelet nodes share the same dimensionality at the furthest depth. This step ensures coherence in the reduced embeddings. They may be extracted from different wavelets but they have the same dimension. Finally, the information from these identified nodes, encompassing basis vectors, indices, and scaling factors is aggregated to construct the embeddings matrix representing the intrinsic, lower-dimensional features associated with each dataset point at the lowest consistent scale.

It is important to note that it is possible to walk up or further down the wavelet tree and extract alternative embeddings based on a chosen scale. For this project, we opted to choose the best scale dynamically based on the deepest level of the tree where all points are represented at the same dimension.

We did observe situations in this project where the deepest such level consisted of embeddings either close to the leaves of the tree, where there were only two nodes per embedding (we saw this mostly with synthetic input data, such as a point cloud sampled from a sphere in high dimensional space) or at the root, where the reduced dimension can be equal to the original dataset size. In these scenarios, the embedding dimensions diverged very quickly after basic clustering. This was not the case for the MNIST data or the LANL data for which we will discuss results for in this chapter but we did observe it with some cyber datasets that we analyzed with GMRA for CASTLE. In these scenarios, we also experimented with an algorithm to explore further down the tree asymmetrically, and find the most reduced common dimension between wavelets nodes whose union contains the most points from the original dataset, even if they are not at the same level in the wavelet tree and not all points are represented.

7.5.3 Batch Processing

In order to mitigate the expense of inserting many nodes into a cover tree at once, we implemented an option for batch processing, which was not in the original GMRA workflow. Each point in the dataset is processed and inserted into the cover tree data structure sequentially, maintaining the tree's integrity throughout the insertion process. We begin by initializing the cover tree with parameters such as maximum scale and base value. The insert method handles point insertion by iterating through each point in the dataset, either initializing the tree with the first dataset or appending subsequent datasets to the existing structure. The results we experimentally found showed exact replication between points inserted in batches vs points inserted all at once, but the batch processing allows for generating results at intermediate steps or adding in new points later in an online process without having to recompute the entire cover tree. This batch processing step is essentially an optimization to the origianl GMRA algorithm, but it does not change the ineherent behavior.

7.5.4 Inverse Wavelet Reconstruction

We successfully implemented the reconstruction of the reduced GMRA embeddings back to the original embedding size. This process was discussed in Allard's original paper [2], but no python implementation existed. Although this step was not required for our analysis, we decided to implement it for completion and also for verification of our embeddings code. If we show that we are able to accurately reconstruct the data, then that provides credence to our implementation. For verification, we focused specifically on images from the MNIST dataset because they are easier to verify than the cyber graph embeddings.

The forward dimension reduction algorithm computes wavelet coefficients for each data point in the lower-dimensional wavelet basis. The wavelet tree structure contains all of the data to extract embeddings, such as wavelet coefficients and bases for each wavelet. The core of the reconstruction process involves traversing the wavelet tree bottom-up, from leaf nodes to the root. At each level of the tree or scale, we accumulate projections from the wavelet bases up the tree to reconstruct the original data.

We evaluated this inverse process on th MNISt dataset, a famous machine learning dataset that contains hand written digit images for classification tasks. We were able to extract reduced dimensions of size 11 with GMRA from the original 28x28 image (which, when flattened became a point with dimension 784). This reduction size makes a lot of sense, because there are 10 classes in the MNIST dataset, so it is very reassuring that GMRA found this intrinsic dimension that intuitively tracks. We show the results of this inverse reconstruction back from the 11 dimension embeddings for sample digits in Figure 7.2. Most of the reconstructed 784 dimensions are extremely coherent as their original digits, with only some noise added from the reconstruction. This inverse code verification process with the MNIST dataset ensures that the lower-dimensional vector embeddings are accurately computed and can be reconstructed with high fidelity.

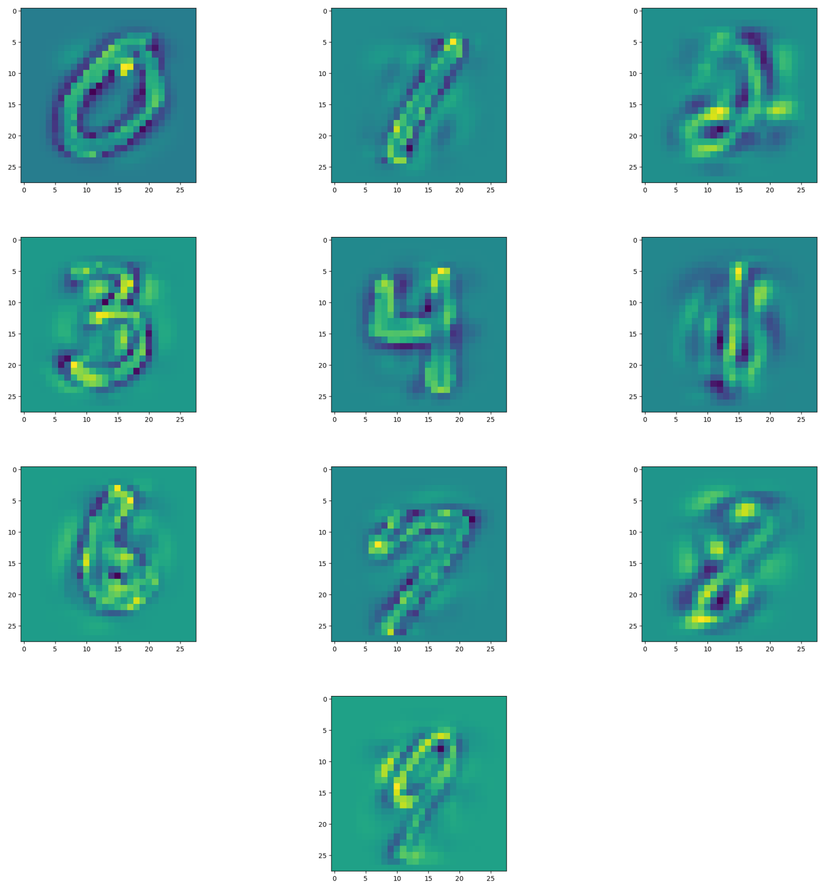


Figure 7.2: This figure demonstrates the bidirectional verification method applied to MNIST digits [0-9]. By computing the inverse of the GMRA reduced embeddings, back into higher-dimensional spaces, we ensure precise representations of data in both datasets, affirming the accuracy of calculations.

7.6 Experimental Results

This research investigates the application of Geometric Multi-Resolution Analysis (GMRA) on graph data and specifically cyber network analysis with the LANL dataset, presenting a novel approach to representing nodes within a network structure. By leveraging GMRA, it becomes possible to compute low-dimensional representations of these graphs and networks, facilitating more efficient processing and analysis.

The LANL dataset contains ground truth for malicious source nodes and their communication with the rest of the network. We tested the impact of the reduced dimensions from GMRA on supervised and unsupervised learning tasks in graph processing, and compare them with the same methods trained on original embeddings from traditional methods like Node2Vec. Our experimental findings indicate that classifiers trained on GMRA embeddings achieve comparable accuracy to those trained on Node2Vec embeddings, even after significant dimensionality reduction by GMRA for two tasks, node classification and anomaly detection.

7.6.1 Node Classification

Node classification is a prevalent machine learning task in graph data analysis. In this context, we used supervised methods with an 80/20 percent split of train and test data. A vast majority of the data was non-malicious, leading to a very skewed training process limited by the quality of the data.

We tested six off-the-shelf classification methods from the sklearn python library with the LANL data, using the original 256 dimension Node2Vec embeddings and their reduced dimension forms from GMRA. The intrinsic dimension determined by GMRA via the dimension at the deepest level of the wavelet tree where all nodes had the same dimension was 31. We chose to test a variety of classification models in order to demonstrate high performance with as many techniques as possible.

- **Nearest Neighbors**: Nearest Neighbors is a simple algorithm that classifies data points based on the majority class of their nearest neighbors. It relies on the assumption that similar data points belong to the same class. The class of a new data point is determined by the class labels of its nearest neighbors in the training data.

- **Decision Tree**: Decision Tree is a tree-like structure where each internal node represents a feature, each branch represents a decision based on that feature, and each leaf node represents a class label. It recursively partitions the feature space into regions, making decisions based on the values of features to classify data points.

- **Neural Net**: Neural Network is a computational model inspired by the structure and function of the human brain. In the context of classification, it consists of interconnected layers of neurons that transform input data into useful representations and make predictions. Through training, neural networks learn to adjust their parameters to minimize prediction errors. In this case, the neural net is a simple 2 layer fully connected construction.

- **AdaBoost**: AdaBoost (Adaptive Boosting) is an ensemble learning technique that combines multiple weak classifiers to create a strong classifier. It sequentially trains a series of weak classifiers on various subsets of the training data, assigning higher weights to misclassified instances in subsequent iterations. The final classification is determined by a weighted sum of the weak classifiers' predictions.

- **Naive Bayes**: Naive Bayes is a probabilistic classification algorithm based on Bayes' theorem and the assumption of conditional independence between features. It calculates the probability of each class given a set of features and selects the class with the highest probability. Despite its simplicity, Naive Bayes often performs well on text classification and other tasks with high-dimensional data.

- **QDA**: Quadratic Discriminant Analysis is a classification algorithm that models the distribution of each class using quadratic decision boundaries. Unlike Naive Bayes, QDA does not assume equal covariance matrices for all classes, allowing it to capture more complex relationships between features. QDA estimates class probabilities based on the multivariate normal distribution of features.

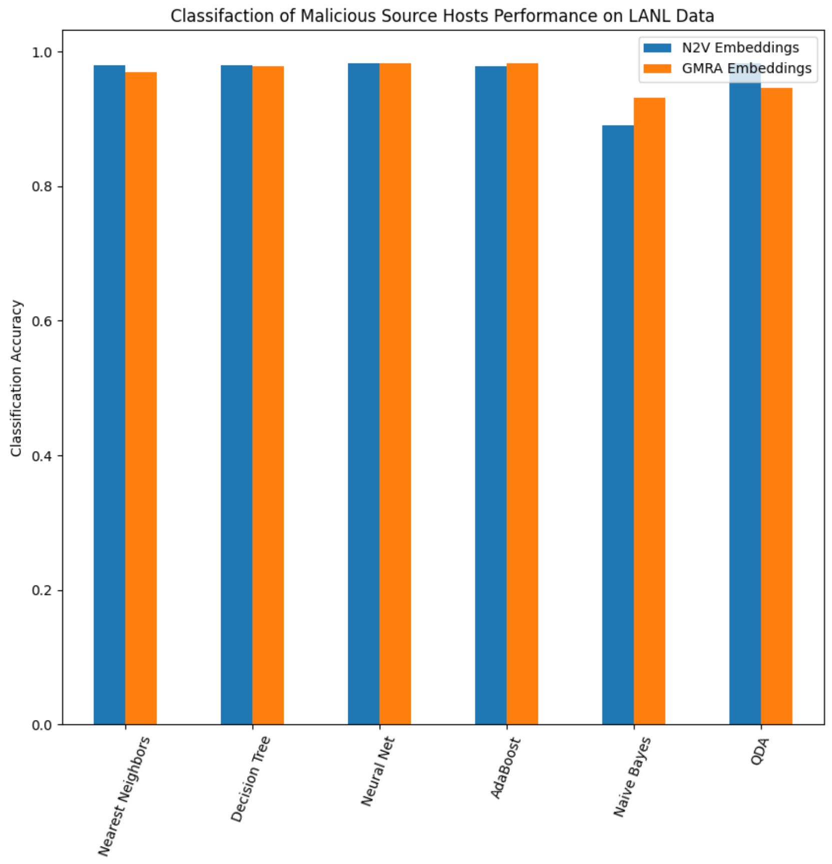


Figure 7.3: Analysis of Classification Methods on the LANL Dataset Using Original and GMRA-Reduced Embedding Vectors.

The accuracy score was computed using the sklearn metrics accuracy score. The comparable accuracy scores between the original and GMRA reduced datasets indicate consistent model performance across both sets of datasets even after a heavy reduction from 256 dimension to 31 dimension, as seen in Figure 7.3. This demonstrates that the geometric wavelet embeddings removed extraneous or repeated data, maintaining the minimum required information to still represent the network's point cloud in a lower dimension.

It is important to note that due to the large amount of negative data and small amount of positive data for training and testing, the recall was approximately 10% on average for node classification, and most of the high accuracy score is due to the correct classification of benign hosts. However, this performance is consistent between the original and reduced node embeddings and thus still supports evidence that GMRA is a useful tool for graph representations.

7.6.2 Anomaly Detection

Anomaly detection is an unsupervised task to detect events, behavior, or entities that are outside the established norm. In the context of cyber networks, this involves identifying malicious node activity. We again used off the shelf anomaly detection using the embeddings from GMRA and Node2Vec for LANL as the input to the anomaly detection system, as an unsupervised counterpart to the node classification method. We tested anomaly detection with the following four methods from sklearn in Python:

Method	Original Dimension	Original Accuracy	Reduced Dimension	Reduced Accuracy
Robust Covariance	256	0.9511	31	0.9481
One-Class SVM	256	0.9490	31	0.9463
Isolation Forest	256	0.9479	31	0.9461
Local Outlier Factor	256	0.9517	31	0.9468

Table 7.1: Anomaly detection results on the LANL cyber network Dataset

- **Robust covariance**: Robust Covariance is an anomaly detection method that estimates the covariance matrix of the data while downweighting the influence of outliers. It is particularly useful when dealing with datasets that contain outliers or are not normally distributed. By robustly estimating the covariance matrix, it identifies anomalies as data points that deviate significantly from the learned covariance structure.

- **One-Class SVM**: One-Class SVM is a machine learning algorithm that learns a decision boundary around the normal instances in a dataset. It constructs a hyperplane in the feature space that encloses the majority of the data points, considering them as the "normal" class. Anomalies are identified as data points lying outside this boundary. One-Class SVM is effective in scenarios where only normal data is available for training.

- **Isolation Forest**: Isolation Forest is an ensemble-based anomaly detection technique that isolates anomalies by randomly partitioning the data space using decision trees. Unlike traditional methods that try to separate normal and abnormal instances, Isolation Forest focuses on isolating anomalies by exploiting the fact that they are typically few in number and have attribute-values that are significantly different from normal instances. This makes it efficient and effective, especially for high-dimensional datasets.

- **Local Outlier Factor**: Local Outlier Factor is a density-based anomaly detection algorithm that measures the local deviation of a data point with respect to its neighbors. It computes the density of data points around each instance and compares it to the density of its neighbors. Anomalies are identified as data points with significantly lower densities compared to their neighbors. LOF is effective in detecting outliers in datasets with varying densities or clusters.

The accuracy score was again computed using sklearn metrics accuracy score. The comparable accuracy scores between the original and GMRA reduced datasets indicate consistent model performance across both sets of datasets. Again we were able to reduce the LANL Dataset Embedding Vectors from 256 dimensions to an intrinsic 31 dimensions, as demonstrated in the previous result of node classification. The high accuracy score

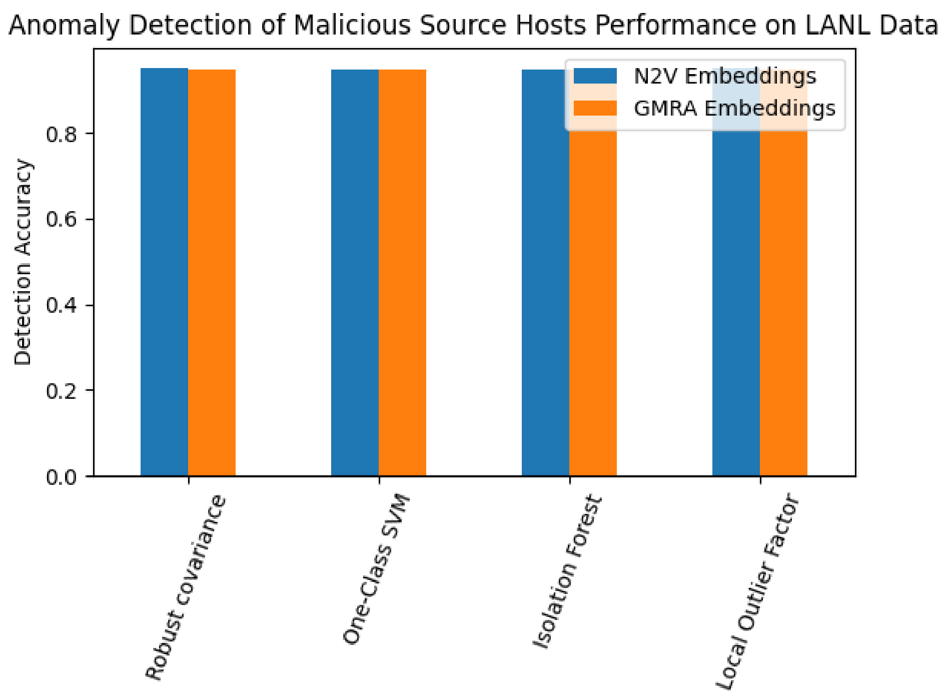


Figure 7.4: Analysis of Anomaly Detection Methods on the LANL Dataset Using Original and GMRA-Reduced Embedding Vectors.

indicates accurate predictions for the majority of samples in both datasets, signifying a strong generalization of the model to both the original and GMRA reduced datasets, as shown in Table 7.1.

Compared to node classification, anomaly detection results showed similar performance, with a notable distinction being the number of false positives. Anomaly detection had much higher recall than node classification with lower precision due to the increased number of false positives. The specific counts of false positives compared to true positives are shown in Table 7.2. To improve this result, we likely need to improve the feature set by adding additional data beyond simply the topology of the network represented as embeddings. It would be advantageous to include data related to event types in addition to counts, which is currently not represented in the input graph embedding data. However, the main take-away of this result is how well the GMRA embeddings capture the information from the original graph embeddings in a much smaller dimension, saving on space and time com-

Method	Original Dimension	Original False Positives	Original True Positives	Reduced Dimension	Reduced False Positives	Reduced True Positives
Robust Covariance	256	218	53	31	230	41
One-Class SVM	256	226	45	31	239	32
Isolation Forest	256	232	39	31	240	31
Local Outlier Factor	256	215	56	31	237	34

Table 7.2: Anomaly detection false positive rates on the LANL cyber network Dataset

plexity. From this perspective, we can see that the results are extremely similar between the two embedding schemes, demonstrating the power of the GMRA technique once again on graph data.

7.7 Conclusion

In this chapter we have proposed a novel technique of applying GMRA for dimensionality reduction in graph embeddings. By demonstrating comparable performance to its higher-dimensional counterparts for various classical graph tasks including node classification, edge classification, graph classification, link prediction, anomaly detection, and graph clustering, it validates the efficacy of GMRA and underscores its efficiency in reducing computational complexity without sacrificing predictive power for downstream processes.

This study proves the potential of leveraging dimensionality reduction techniques like GMRA to address the challenges posed by high-dimensional data in graph based tasks. Additionally, it sheds light on several avenues for future exploration and enhancement:

1. **Fine-tuning GMRA Parameters:** Further investigation into optimal parameter settings of GMRA, such as the choice of clustering function and hyperparameters based

on the specific graph task.

2. **Exploring Additional Graph Tasks:** Such examples could potentially include graph classification and cross network alignment to provide evidence for broader applicability.

3. **Improving Scalability:** Currently, the GMRA algorithm is inefficient on graphs with a very large number of nodes due to the covertree algorithm. There is an opportunity here for improving scalability of the graph via optimization or investigating alternative clustering techniques.

4. **Interpretability and Visualization:** Developing techniques to interpret and visualize the transformed embeddings obtained through GMRA could provide insights into the underlying structures and relationships present within complex network data.

Chapter 8

Conclusion

To summarize, in this thesis we have outlined six projects that are linked by their connection to the theme of machine learning applied to graph theory tasks, with an emphasis on graph representations. Each chapter consisted of a complete review of the research project, encompassing specific background dedicated to the problem, a method outline, experimental results, interpretation, and concluding thoughts with potential future work discussions.

As a review, the chapters covered domains such as distance labeling, shortest path distance approximation, distance sensitivity oracle, stochastic block model, cross-network alignment, and node embedding dimensionality reduction. Collectively, these six projects provide a comprehensive survey of novel machine learning methods applied to practical graph problems.

The main theme throughout the paper is the importance of the choice of graph representation for graph tasks. We experimented with many different state-of-the-art graph represenations, including Node2Vec [20], GraRep [25], and GraphSage [29], and also discussed various custom ways to create node embeddings for specific tasks, such as in Chapters 1-3.

In the final chapter we also covered dimensionality reduction of graph embeddings for downstream graph tasks, and how performance on a wide variety of problems can be maintained even with lower dimensional representations of general graph embeddings.

The totality of the findings outlined in this thesis underscores the significance of thoughtful representation choices in optimizing the performance and applicability of machine learning models to graph-related tasks.

Bibliography

[1] Yutao Zhang, Jie Tang, Zhilin Yang, Jian Pei, and Philip S. Yu. Cosnet: Connecting heterogeneous social networks with local and global consistency. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, page 1485â1494, New York, NY, USA, 2015. Association for Computing Machinery.

[2] William K Allard, Guangliang Chen, and Mauro Maggioni. Multi-scale geometric methods for data sets ii: Geometric multi-resolution analysis. *Applied and computational harmonic analysis*, 32(3):435–462, 2012.

[3] Robin J Wilson. *Introduction to graph theory*. John Wiley & Sons, Inc., USA, 1986.

[4] Edsger W Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.

[5] Hongyun Cai, Vincent W Zheng, and Kevin Chen-Chuan Chang. A comprehensive survey of graph embedding: Problems, techniques, and applications. *IEEE Transactions on Knowledge and Data Engineering*, 30(9):1616–1637, 2018.

[6] Xianchao Zhang, Jie Mu, Han Liu, and Xiaotong Zhang. Graphnet: Graph clustering with deep neural networks. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3800–3804. IEEE, 2021.

[7] Gamal Crichton, Yufan Guo, Sampo Pyysalo, and Anna Korhonen. Neural networks for link prediction in realistic biomedical graphs: a multi-dimensional evaluation of graph embedding-based approaches. *BMC bioinformatics*, 19(1):1–11, 2018.

[8] Shuai Huang, Yong Wang, Tianyu Zhao, and Guoliang Li. A learning-based method for computing shortest path distances on road networks. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, pages 360–371. IEEE, 2021.

[9] Yuanfang Ren, Ahmet Ay, and Tamer Kahveci. Shortest path counting in probabilistic biological networks. *BMC bioinformatics*, 19(1):1–19, 2018.

[10] Xiuxia Tian, Yangli Song, Xiaoling Wang, and Xueqing Gong. Shortest path based potential common friend recommendation in social networks. In *2012 Second International Conference on Cloud and Green Computing*, pages 541–548. IEEE, 2012.

[11] Antti Ukkonen, Carlos Castillo, Debora Donato, and Aristides Gionis. Searching the wikipedia with contextual information. In *Proceedings of the 17th ACM conference on Information and knowledge management*, pages 1351–1352, 2008.

[12] Adam Breuer, Roee Eilat, and Udi Weinsberg. Friend or faux: Graph-based early detection of fake accounts on social networks. *CoRR*, abs/2004.04834, 2020.

[13] Alexander D. Kent. Comprehensive, multi-source cyber-security events data set. 5 2015.

[14] Xiao-Meng Zhang, Li Liang, Lin Liu, and Mingjing Tang. Graph neural networks and their current applications in bioinformatics. *Frontiers in Genetics*, 12, 2021.

[15] Wayne Zachary. An information flow model for conflict and fission in small groups1. *Journal of anthropological research*, 33, 11 1976.

[16] Mert Ozer, Nyunsu Kim, and Hasan Davulcu. Community detection in political twitter networks using nonnegative matrix factorization methods. In *2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 81–88, 2016.

[17] Yan Liu, Alexandru Niculescu-Mizil, and Wojciech Gryc. Topic-link lda: joint models of topic and author community. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, page 665â672, New York, NY, USA, 2009. Association for Computing Machinery.

[18] Linyuan LÃ¼ and Tao Zhou. Link prediction in complex networks: A survey. *Physica A: Statistical Mechanics and its Applications*, 390(6):1150–1170, 2011.

[19] 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.

[20] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks, 2016.

[21] Leonardo F.R. Ribeiro, Pedro H.P. Saverese, and Daniel R. Figueiredo. struc2vec. *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Aug 2017.

[22] Bryan Perozzi, Vivek Kulkarni, Haochen Chen, and Steven Skiena. Don't walk, skip! online learning of multi-scale network embeddings, 2017.

[23] Haochen Chen, Bryan Perozzi, Yifan Hu, and Steven Skiena. Harp: Hierarchical representation learning for networks, 2017.

[24] Amr Ahmed, Nino Shervashidze, Shravan Narayanamurthy, Vanja Josifovski, and Alexander J. Smola. Distributed large-scale natural graph factorization. In *Proceedings of the 22nd International Conference on World Wide Web*, WWW '13, page 37â48, New York, NY, USA, 2013. Association for Computing Machinery.

[25] Shaosheng Cao, Wei Lu, and Qiongkai Xu. Grarep: Learning graph representations with global structural information. In *Proceedings of the 24th ACM international on conference on information and knowledge management*, pages 891–900, 2015.

[26] Mingdong Ou, Peng Cui, Jian Pei, Ziwei Zhang, and Wenwu Zhu. Asymmetric transitivity preserving graph embedding. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 1105â1114, New York, NY, USA, 2016. Association for Computing Machinery.

[27] Jie Zhang, Yuxiao Dong, Yan Wang, Jie Tang, and Ming Ding. Prone: Fast and scalable network representation learning. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 4278–4284. International Joint Conferences on Artificial Intelligence Organization, 7 2019.

[28] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009.

[29] William L. Hamilton, Zhitao Ying, and Jure Leskovec. Inductive Representation Learning on Large Graphs. In *NIPS*, pages 1024–1034, 2017.

[30] Ikeoluwa Abioye, Allison Gunby-Mann, Xu Wang, Sarel Cohen, and Sang Chin. *Learned Approximate Distance Labels for Graphs*, pages 339–350. 02 2024.

[31] Frank Pacini, Allison Gunby-Mann, Sarel Cohen, and Sang Chin. Aneda: Adaptable node embeddings for shortest path distance approximation. pages 1–7, 09 2023.

[32] Davin Jeong, Allison Gunby-Mann, Sarel Cohen, Maximilian Katzmann, Chau Pham, Arnav Bhakta, Tobias Friedrich, and Sang Chin. Deep distance sensitivity oracles, 2023.

[33] Stephen Alstrup, Philip Bille, and Theis Rauhe. Labeling schemes for small distances in trees. *SIAM Journal on Discrete Mathematics*, 19(2):448–462, 2005.

[34] Stephen Alstrup, Inge Li Gørtz, Esben Bistrup Halvorsen, and Ely Porat. Distance labeling schemes for trees. In *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016*, volume 55 of *LIPIcs*, pages 132:1–132:16, 2016.

[35] Ofer Freedman, Paweł Gawrychowski, Patrick K. Nicholson, and Oren Weimann. Optimal distance labeling schemes for trees. In *Proceedings of the ACM Symposium on Principles of Distributed Computing*, PODC '17, page 185â194, 2017.

[36] Cyril Gavoille, David Peleg, StÃ©phane PÃ©rennes, and Ran Raz. Distance labeling in graphs. *Journal of Algorithms*, 53(1):85–112, 2004.

[37] Stephen Alstrup, SÃ¸ren Dahlgaard, Mathias BÃ¦k Tejs Knudsen, and Ely Porat. Sublinear Distance Labeling. In *24th Annual European Symposium on Algorithms (ESA 2016)*, volume 57, pages 5:1–5:15, 2016.

[38] Stephen Alstrup, Haim Kaplan, Mikkel Thorup, and Uri Zwick. Adjacency labeling schemes and induced-universal graphs. In *Proceedings of the Forty-Seventh Annual ACM Symposium on Theory of Computing*, STOC '15, page 625â634, 2015.

[39] Fatemeh Salehi Rizi, Joerg Schloetterer, and Michael Granitzer. Shortest path distance approximation using deep learning techniques. In *2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 1007–1014. IEEE, 2018.

[40] Jianzhong Qi, Wei Wang, Rui Zhang, and Zhuowei Zhao. A learning based approach to predict shortest-path distances. In *EDBT*, 2020.

[41] Dustin Brunner. Distance preserving graph embedding, 2021.

[42] Hang-Sheng Wang, Xinping Zhu, Li-Shiuan Peh, and Sharad Malik. Orion: A power-performance simulator for interconnection networks. In *35th Annual IEEE/ACM International Symposium on Microarchitecture, 2002.(MICRO-35). Proceedings.*, pages 294–305. IEEE, 2002.

[43] Andrej Cvetkovski and Mark Crovella. Hyperbolic embedding and routing for dynamic graphs. In *IEEE INFOCOM 2009*, pages 1647–1655. IEEE, 2009.

[44] Xiaohan Zhao, Alessandra Sala, Haitao Zheng, and Ben Y Zhao. Efficient shortest paths on massive social graphs. In *7th International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom)*, pages 77–86. IEEE, 2011.

[45] Ishan Jindal, Xuewen Chen, Matthew Nokleby, Jieping Ye, et al. A unified neural network approach for estimating travel time and distance for a taxi trip. *arXiv preprint arXiv:1710.04350*, 2017.

[46] Jianzhong Qi, Wei Wang, Rui Zhang, and Zhuowei Zhao. A learning based approach to predict shortest-path distances. In *EDBT*, pages 367–370, 2020.

[47] Xu Chen, Shaohua Wang, Huilai Li, Fangzheng Lyu, Haojian Liang, Xueyan Zhang, and Yang Zhong. Ndist2vec: Node with landmark and new distance to vector method for predicting shortest path distance along road networks. *ISPRS International Journal of Geo-Information*, 11(10):514, 2022.

[48] Ryan A. Rossi and Nesreen K. Ahmed. The network data repository with interactive graph analytics and visualization. In *AAAI*, 2015.

[49] Geoff Boeing. Osmnx: New methods for acquiring, constructing, analyzing, and visualizing complex street networks. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, volume 65, pages 126–139, 2017.

[50] Alireza Karduni, Amirhassan Kermanshah, and Sybil Derrible. A protocol to convert spatial polyline data to network formats and applications to world urban road networks. volume 3. Scientific Data, 2016.

[51] Jure Leskovec and Andrej Krevl. Snap datasets: Stanford large network dataset collection, 2014.

[52] Aditya Grover and Jure Leskovec. Node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 855â864, New York, NY, USA, 2016. Association for Computing Machinery.

[53] Puoya Tabaghi and Ivan Dokmanić. Hyperbolic distance matrices. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1728–1738, 2020.

[54] Camil Demetrescu, Mikkel Thorup, Rezaul Alam Chowdhury, and Vijaya Ramachandran. Oracles for distances avoiding a failed node or link. *SIAM J. Comput.*, 37(5):1299–1318, 2008.

[55] Aaron Bernstein and David R. Karger. A nearly optimal oracle for avoiding failed vertices and edges. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pages 101–110. ACM, 2009.

[56] Andrea Lincoln, Virginia Vassilevska Williams, and R. Ryan Williams. Tight hardness for shortest cycles and paths in sparse graphs. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, pages 1236–1252. SIAM, 2018.

[57] Shiri Chechik and Sarel Cohen. Distance sensitivity oracles with subcubic preprocessing time and fast query time. In *Proccedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, June 22-26, 2020*, pages 1375–1388. ACM, 2020.

[58] Hanlin Ren. Improved distance sensitivity oracles with subcubic preprocessing time. *J. Comput. Syst. Sci.*, 123:159–170, 2022.

[59] Yong Gu and Hanlin Ren. Constructing a Distance Sensitivity Oracle in $O(n^2.5794M)$ Time. In Nikhil Bansal, Emanuela Merelli, and James Worrell, editors, *48th International Colloquium on Automata, Languages, and Programming (ICALP 2021)*, volume 198 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 76:1–76:20, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[60] Ran Duan and Tianyi Zhang. Improved distance sensitivity oracles via tree partitioning. In *Algorithms and Data Structures - 15th International Symposium, WADS 2017, St. John's, NL, Canada, July 31 - August 2, 2017, Proceedings*, volume 10389 of *Lecture Notes in Computer Science*, pages 349–360. Springer, 2017.

[61] Surender Baswana and Neelesh Khanna. Approximate Shortest Paths Avoiding a Failed Vertex: Near Optimal Data Structures for Undirected Unweighted Graphs. *Algorithmica*, 66:18–50, 2013.

[62] Shiri Chechik, Michael Langberg, David Peleg, and Liam Roditty. f-Sensitivity Distance Oracles and Routing Schemes. *Algorithmica*, 63:861–882, 2012.

[63] Yehuda Afek, Anat Bremler-Barr, Haim Kaplan, Edith Cohen, and Michael Merritt. Restoration by path concatenation: fast recovery of MPLS paths. *Distributed Comput.*, 15(4):273–283, 2002.

[64] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[65] Thomas Bläsius, Tobias Friedrich, Maximilian Katzmann, and Anton Krohmer. Hyperbolic embeddings for near-optimal greedy routing. *ACM J. Exp. Algorithmics*, 25, 2020.

[66] Peter Chin, Anup Rao, and Van Vu. Stochastic block model and community detection in sparse graphs: A spectral algorithm with optimal rate of recovery. In Peter GrÃ¼nwald, Elad Hazan, and Satyen Kale, editors, *Proceedings of The 28th Conference on Learning Theory*, volume 40 of *Proceedings of Machine Learning Research*, pages 391–423, Paris, France, 03–06 Jul 2015. PMLR.

[67] Amin Coja-Oghlan. Graph partitioning via adaptive spectral techniques. *Combinatorics, Probability and Computing*, 19(2):227â284, 2010.

[68] Emmanuel Abbe. Community detection and stochastic block models, 2023.

[69] Soumik Pal and Yizhe Zhu. Community detection in the sparse hypergraph stochastic block model. *Random Structures & Algorithms*, 59(3):407â463, March 2021.

[70] Kevin S. Xu and Alfred O. Hero. Dynamic stochastic blockmodels for time-evolving social networks. *IEEE Journal of Selected Topics in Signal Processing*, 8(4):552â562, August 2014.

[71] Chandler Davis. The rotation of eigenvectors by a perturbation. *Journal of Mathematical Analysis and Applications*, 6(2):159–173, 1963.

[72] Rajendra Bhatia. *Matrix Analysis*, volume 169. Springer, 1997.

[73] J. Friedman, J. Kahn, and E. Szemerédi. On the second eigenvalue of random regular graphs. In *Proceedings of the Twenty-First Annual ACM Symposium on Theory of Computing*, STOC '89, page 587â598, New York, NY, USA, 1989. Association for Computing Machinery.

[74] Uriel Feige and Eran Ofek. Spectral techniques applied to sparse random graphs. *Random Structures & Algorithms*, 27(2):251–275, 2005.

[75] Svante Janson, Tomasz Luczak, and Andrzej Rucinski. *Theory of random graphs*. John Wiley & Sons, New York; Chichester, 2000.

[76] Mark Heimann, Haoming Shen, Tara Safavi, and Danai Koutra. Regal: Representation learning-based graph alignment. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, CIKM â18. ACM, October 2018.

[77] Si Zhang and Hanghang Tong. Final: Fast attributed network alignment. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 1345â1354, New York, NY, USA, 2016. Association for Computing Machinery.

[78] Rohit Singh, Jinbo Xu, and Bonnie Berger. Global alignment of multiple protein interaction networks with application to functional orthology detection. *Proceedings of the National Academy of Sciences*, 105(35):12763–12768, 2008.

[79] Chung-Yi Li and Shou-De Lin. Matching users and items across domains to improve the recommendation quality. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, page 801â810, New York, NY, USA, 2014. Association for Computing Machinery.

[80] Danai Koutra, Hanghang Tong, and David Lubensky. Big-align: Fast bipartite graph alignment. In *2013 IEEE 13th International Conference on Data Mining*, pages 389–398, 2013.

[81] Kaikai Deng, Ling Xing, Longshui Zheng, Honghai Wu, Ping Xie, and Feifei Gao. A user identification algorithm based on user behavior analysis in social networks. *IEEE Access*, 7:47114–47123, 2019.

[82] Francesco Buccafurri, Gianluca Lax, Antonino Nocera, and Domenico Ursino. Discovering links among social networks. In Peter A. Flach, Tijl De Bie, and Nello Cristianini, editors, *Machine Learning and Knowledge Discovery in Databases*, pages 467–482, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[83] Xiaoping Zhou, Xun Liang, Haiyan Zhang, and Yuefeng Ma. Cross-platform identification of anonymous identical users in multiple social media networks. *IEEE Transactions on Knowledge and Data Engineering*, 28(2):411–424, 2016.

[84] Tong Man, Huawei Shen, Shenghua Liu, Xiaolong Jin, and Xueqi Cheng. Predict anchor links across social networks via an embedding approach. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, page 1823â1829. AAAI Press, 2016.

[85] Li Liu, William K. Cheung, Xin Li, and Lejian Liao. Aligning users across social networks using network embedding. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, page 1774â1780. AAAI Press, 2016.

[86] Fan Zhou, Lei Liu, Kunpeng Zhang, Goce Trajcevski, Jin Wu, and Ting Zhong. Deeplink: A deep learning approach for user identity linkage. In *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, pages 1313–1321, 2018.

[87] Oana Goga, Patrick Loiseau, Robin Sommer, Renata Teixeira, and Krishna P. Gummadi. On the reliability of profile matching across large online social networks. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, pages 1799–1808, New York, NY, USA, 2015. ACM.

[88] Oana Goga, Howard Lei, Sree Hari Krishnan Parthasarathi, Gerald Friedland, Robin Sommer, and Renata Teixeira. Exploiting innocuous activity for correlating users across sites. In *Proceedings of the 22Nd International Conference on World Wide Web*, WWW '13, pages 447–458, New York, NY, USA, 2013. ACM.

[89] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013.

[90] Sami Abu-El-Haija, Bryan Perozzi, and Rami Al-Rfou. Learning edge representations via low-rank asymmetric projections. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, CIKM â17. ACM, November 2017.

[91] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

[92] 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*, KDD â14. ACM, August 2014.

[93] William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs, 2018.

[94] Guangliang Chen, Mark Iwen, Sang Chin, and Mauro Maggioni. A fast multiscale framework for data in high-dimensions: Measure estimation, anomaly detection, and compressive measurements. In *2012 Visual Communications and Image Processing*, pages 1–6, 2012.

[95] Yi Wang, Guangliang Chen, and Mauro Maggioni. High-dimensional data modeling techniques for detection of chemical plumes and anomalies in hyperspectral images and movies. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 9(9):4316–4324, 2016.

[96] Hieu Le, Andrew Wood, Sylee Dandekar, and Peter Chin. Neural network optimization with biologically inspired low-dimensional manifold learning. In *2021 International Conference on Computational Science and Computational Intelligence (CSCI)*, pages 8–13, 2021.

[97] Mangalraj Poobalasubramanian, Sivakumar V, Karthick Selvam, V. Haribaabu, Ramraj Santhanam, and Dinesh Jackson Samuel. A review of multi-resolution analysis (mra) and multi-geometric analysis (mga) tools used in the fusion of remote sensing images. *Circuits, Systems, and Signal Processing*, 39, 06 2020.