DISSERTATION

EFFICIENT REPRESENTATION, MEASUREMENT, AND RECOVERY OF SPATIAL AND SOCIAL NETWORKS

Submitted by

Gunjan S. Mahindre

Department of Electrical and Computer Engineering

In partial fulfillment of the requirements

For the Degree of Doctor of Philosophy

Colorado State University

Fort Collins, Colorado

Fall 2021

Doctoral Committee:

Advisor: Anura Jayasumana

Randy Paffenroth
Anthony Maciejewski
Michael Kirby

ABSTRACT

EFFICIENT REPRESENTATION, MEASUREMENT, AND RECOVERY OF SPATIAL AND SOCIAL NETWORKS

Massive penetration of networks such as social and communication networks and the growth of networked systems are leading to a colossal number of connections and data underneath. Techniques are needed for analysis of network measurements to extract network features and patterns of interest. Network analytics can be used, for example, to predict how a network evolves for customer interaction analysis to provide a better user experience, and to make effective business decisions based on client engagements. Ideally, one should have the complete set of measurements associated with a network so that the analysis would lead to results that are applicable to real-world networks. However, computational, communication, and storage limitations do not allow that for large-scale networks of interest. Thus, a network needs to be sampled for its connectivity or substructures to facilitate analysis. However, extensive sampling can be computationally expensive to carry out for analysis or even unviable due to access restrictions, e.g., due to private or censored nodes. Also, currently practiced sampling techniques such as random-walk are susceptible to slow-mixing problem. Sparse or locally targeted sampling techniques do not allow the recovery of the complete information of the original network while missing information may lead to altered network characteristics and biased research results.

The limitations identified above give rise to the need for techniques to correctly predict missing connectivity data as well as to represent and store networks in coherent yet compressed ways that preserve the original network characteristics for unbiased network experiments. As certain network features are derived from the complete network measurements, missing data or local samples affect the accuracy of such estimates. Efficient prediction and graph representation techniques

could facilitate network analytics from smaller sets of measurements with improved accuracy, low storage, ease of access as well as manageable computational costs.

Our goal is to make the most use of the available network measurements and extract information about network characteristics, connectivity, and constraints to make informed predictions about the missing node-pair distances. Accordingly, this research is aimed at deriving and demonstrating novel techniques to efficiently measure and represent networks, and for accurate recovery of network topology from partially measured network structures. The techniques need to be scalable in their computation, and graceful in their degradation with limited measurements.

We tackle the network analysis problem from two sides: a) efficiently representing complete graph data in compact formats and b) predicting missing network measurements using partially observed distances. Specifically, we introduce the novel concepts of reconstruction set and link dimension of graphs for lossless compression and reconstruction. This helps to represent graphs with a small set of path measurements rather than the complete adjacency or distance matrices. We also develop models to learn the network topology via path measurements and explore distance vector properties of the network. This network information is then leveraged to make informed predictions about network connectivity. This helps to retain original network characteristics even while completing graphs from partial data, in terms of links and distances, using deterministic methods such as low-rank matrix completion as well as non-deterministic prediction methods such as neural networks.

We present link dimension, a novel graph dimension based on a subset of nodes for defining distance vectors that completely capture the graph topology. Several definitions for dimensions of a graph exist, such as metric dimension and edge dimension, to identify the least integer 'n' such that there exists a representation of the graph in n dimensions. Link dimension is however the only dimension that captures the complete topology of the graph and allows for its exact reconstruction. We also propose a greedy algorithm to find such defining nodes in the network by identifying the nodes that capture most of the information about nodes as well as the links. Several interesting properties of link dimension are also derived along with the bounds for link dimension for several

graph families. Ability to recover or estimate the network topology from a small fraction of pairwise distances among nodes or even a few random distance measurements is essential when the network measurements are expensive or infeasible. We use low-rank matrix completion to recover topology of spatial networks from a very small fraction of distance measurements. We present results for networks of different shapes, with a range of 20% to 80% of observed measurements. This technique is especially useful in sensor networks which are constrained with respect to their storage and computational capabilities, each node has access to only partial information about the network. It is helpful to know the connectivity, boundaries, and the overall topology of the sensor networks especially in applications such as routing, segmentation, and anchor selection.

Not all networks are embedded in 2D or 3D physical spaces. Graphs such as friendship networks, product graphs, and software module connectivity are non-spatial. Such graphs can be measured in terms of inter-node distances, i.e., connected nodes are said to be at one hop-distance. We leverage triangle inequality as applied to distances on a graph to compute bounds on the missing distance entries and perform bounded matrix completion on directed as well as undirected social networks. The proposed prediction techniques are evaluated for real-world social networks (such as Facebook, Email, Collaboration networks, Twitter). The low-rank matrix completion based prediction technique is evaluated for 20% to 80% missing node-pair distances in a given network. Results for low-rank matrix completion show that even at 40% of missing node-pair distances, the network distances can be accurately predicted while preserving the original network characteristics.

Many real-world networks are known to exhibit scale-free phenomenon, where the degree distribution follows the power law, at least asymptotically. In order to learn this complex relationship between node distances, we turn to neural networks as a foundation of a new prediction model. However, neural networks need complete data to train on which is not available when only a fraction of distances is measured in a network. We use the concepts of domain adaptation to employ a novel technique of using synthetic networks to pre-train the autoencoder. We make use of the scale free phenomenon observed in real-world networks and generate artificial networks that

also embody this power law in their degree distribution. Training on these preferentially attached synthetic networks helps in learning about the scale-free networks, even if the training data was not derived from the real-world networks. This helps to make accurate predictions on real-world networks when only ultra-sparse measurements are observed. This aids in two ways: a) we can generate ample amounts of training data and b) we can make sure that the training data is similar in characteristics with the real-world network we want to eventually predict on. This helps to improve the prediction accuracy of our non-deterministic approach over the deterministic one when the fraction of observed measurements decreases. The neural network-based model achieves efficient prediction performance with graceful degradation over ultra-sparsely sampled networks. While Low-rank Matrix Completion predicts missing network distances within 15% of error when 20% of distances are missing, neural network-based model gives an estimation error of within 20% when 90% of entries are missing.

In summary, the lossless graph representation can aid faster processing, compact storage, and novel solutions for networking algorithms. Conversely, the ability to predict missing connectivity information from a relatively small set of distance measurements enables inference of hidden connectivity information and provides a foundation for developing novel network mining algorithms.

DEDICATION

I dedicate this thesis to my family, people who always believed in me.

ACKNOWLEDGEMENTS

I would like to thank, from the bottom of my heart, everyone who contributed to this work directly and indirectly. I would not have done this without the support and encouragement of so many amazing people in my life.

I cannot thank Professor Jayasumana enough for his guidance at every step in this journey, for being patient and teaching me how to fish instead of handing me the fish, for countless research meetings, and always being ok with my questions and suggestions. He has taught me how to think. I want to thank Randy for being a mentor, for always being available to answer my doubts and explain difficult concepts with great ease. It is from him, that I learnt how to explain. Randy once told me that we become like our advisors, and if I become even 10% as good as the combination of Randy and Anura, I would be very proud of myself. I would also like to thank Dr. Maciejewski and Dr. Kirby for their guidance and support.

I would like to thank the backbone of my support system, my family. My father (Baba), mother (Aai), Aniket (Dada) and Shivani. I cannot thank Lajos enough for everything he does. I am grateful to have Aniko and Milan who are always cheering for me. You all have always been there for me. Tough times are easier and celebrations are merrier with you all.

My lab mates have been there to talk to me about research as well as life and I feel fortunate to have received their company.

I would like to thank the CSU Graduate Student for supporting this project.

TABLE OF CONTENTS

Chapter 1

Introduction

Social networks are growing rapidly with the increased ease of accessing the Internet. Every minute of the day, Uber gives around 700 rides, Skype users make 110k calls, and Youtube users upload 300 hours of video content [1]. With the growing connections and data underneath, it is becoming more and more important to analyze such network structures. This gives us the motivation to manage such large networks, extract meaningful insights, and efficiently gain an understanding of the interconnections among components which can consequently help us to make more informed decisions. However, with more data comes a few challenges, such as, measurement complexity, storage, processing, and accessing data from storage with low latency.

Networks can be represented as graphs where nodes can be users in a network like LinkedIn and link can be the connection between any two given users. networks an be spatial (such as road transportation network where nodes are places and links are paths between places) or non-spatial (such as Facebook network with users as nodes and friendships between users as links). The distance between two nodes can be calculated as the Euclidean distance in spatial networks in the given coordinate system. However, this is not applicable to non-spatial networks but they can be measured in terms of hops. Two nodes are said to be one hop apart if there is a direct link between them. For example, if two individuals are directly connected on LinkedIn, they would be one hop apart from each other and if they are connected via a common friend and are not directly connected, they would be two hops apart from each other. Even when we know the adjacency information of all nodes, there can be multiple ways of traveling from node A to node B. Thus, distance from node A to B is usually calculated as the shortest path distance, i.e., the minimum number of hops required to take from node A to reach node B. Shortest path lengths can be calculated by algorithms such as Dijkstra's, Bellman-Ford, Depth First Search (DFS), and Breadth First Search (BFS) which have computational complexity of at least $\mathcal{O}(e+n)$ or $\mathcal{O}(e+nlog(n))$ where e is the number of edges and n is the number of nodes in the given network. As the real-world networks today

can be as large as millions and billions of nodes with trillions of edges, the shortest hop-distance between all node pairs may not be available due to limitations in time, accessibility, or resources. Several network analysis applications - routing, topology extraction, node separation, centrality computation, mutual friend detection, link prediction, community detection - rely on computing these shortest node-pair distances. Thus, our focus stays on using shortest node pair distances as our network data throughout this research while utilizing the known measurements and predicting the unknown shortest node pair distances.

In this chapter, we address the influence of network analysis and current challenges that motivate us towards this research in Section 1.1. Section 1.2 introduces the reader to the main contributions of this work and proposed approach towards designing efficient solutions for existing challenges. Section 1.3 explains the impact of this work with a few real-world examples and Section 1.4 presents the outline of this dissertation.

1.1 Motivation

Various social factors such as weather conditions, economy, human behavior, and geography interact when they participate in networks. For instance, a retail network with nodes as shops and their sales data and links as distances between shops are affected by average annual income of people in the nearby area, weather condition, etc. Similarly, an airline path network can tell us about the popular destinations, frequency of travelers, off-seasons, and remote locations. When one or more factors change, we should be informed about its effect on the airline network so that appropriate measures can be taken. For instance, during holiday season airlines can make adequate number of flights available, lower fares for less popular destinations can be offered to encourage travel, or keep passengers well informed if the weather conditions are adverse. This information can be extracted by (1) measuring the network with respect to its nodes (airports) and edges (flight paths), and (2) analyzing the network characteristics and dependencies on different factors. The same applies to studying the spread of contagious diseases, understanding individual behavior in friendship networks, detecting purchase patterns for each user in e-commerce, analyzing seasonal

animal interactions in zoology, etc. Social network analysis has numerous applications. This extends to networks that are spatial, i.e., can be characterized by their geographic localization, such as sensor networks, mobile networks [2], transportation networks [3], the Internet of Things (IoT) [2], power grids, etc. Such networks can be embedded in 2D or 3D physical spaces unlike social networks. Characterizing and studying the topology, structure, and evolution of such networks is crucial in fields like epidemiology, cell communication, and urbanism to name a few.

Graphs represent many underlying structures of real-world networks such as connections among criminals, protein interactions, brain neurons, and networks in general, thus, network analysis through graphs has gained popularity in numerous fields, including computer science, biology, social psychology, demography, communication studies, economics, geography, history, organizational studies, and political science. Network science thus has a range of applications from recommender systems in Netflix, resource allocation, and behavior analysis of social network users [4–6] to interaction analysis of terrorist groups [7]. Business intelligence, security systems, and customer analysis in marketing [8] have also been exploiting network analysis methods to make better decisions.

Though more data may give an illusion of more information, it is not the reality due to several challenging factors in the current network analysis infrastructure:

Challenges:

As researchers typically need at least a subset of nodes and edges to make meaningful conclusions about a complete network, collecting network data plays a foundational role in network analysis [9]. Many network measurement and sampling techniques collect data over the online infrastructure for analysis as it provides access to a great pool of participants, is less expensive, maintains user privacy, and gives an increased sense of control over which information to share [10]. However, network sampling techniques suffer from several limitations which cause problems for accurate network analysis.

Questionnaires, surveys [11], crawling [12], random sampling, snowball sampling, and salting [13, 14] are some of the widely used data collection techniques for social networking applica-

tions. As the online networks can be very large (as of 2016, Facebook has around 1.59 billion active users [15]) complete data collection can be computationally expensive. While network crawling or sampling, because of user anonymity, it may not be possible to follow up with the participants in the case of missing data. Some networks have public and private nodes which restrict access to user information resulting in incomplete data. Thus, the number of users and information about each user often create a trade-off based on how much data is allowed to be accessed. Sampled large graphs, static and especially dynamic graphs are too huge to store because dynamic networks need to be sampled multiple times as they evolve with time. This makes efficient real-time processing extremely difficult. Ego networks, e.g., consist of a focal node ("ego") and the nodes to whom ego is directly connected to (these are called "alters") plus the ties, if any, among the alters [16]. These networks are good to study a subset of a large network when information about a particular phenomenon is desired, however, different participants have different levels of importance depending on the phenomenon to be studied. Ego networks can be oversimplified and thus, may not support community study or network structure as a whole. Snowball sampling [13, 17, 18], a sampling technique in which selected samples are asked to assist in identifying other potential samples and the process is continued like flooding until desired number of samples are collected, is subject to numerous biases such as nodes with large number of connections have more control over the sampling process. This may cause the weaker ties [19] to appear of less influence whereas in actuality it may be the exact opposite.

For many such networks associated with incomplete data collection, the hop-distances acquired or the links observed are only partial entries in the complete distance or adjacency matrix of the graph represented by such networks. As shown in [20], missing data can have a significant impact on the ability to estimate structural properties (clustering coefficient, average path length) of a social network. Study in [21] presents effects of various data removal strategies on characteristics (e.g., proximity index, node, and edge coverage) of multi-layered networks, in which a holistic view of multiple networks is taken. Work done in both the experiments attest that missing or incomplete data changes the representative network significantly to affect its original network

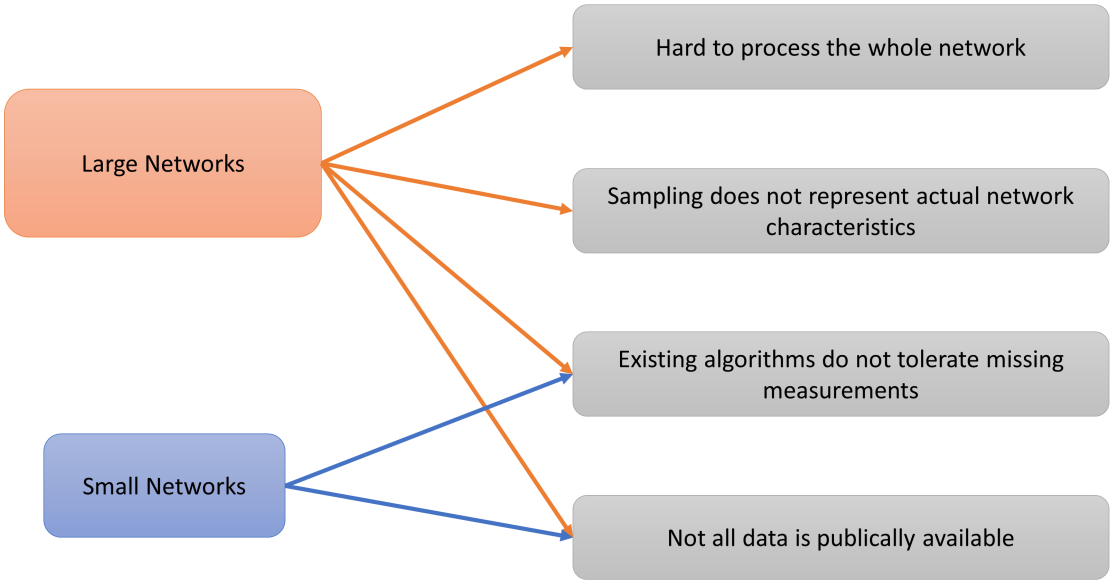


Figure 1.1: Challenges in analyzing large and small networks

or node feature values creating a gap between network data and meaningful network analysis. Estimating the missing entries and completing the available partial data as per the original network characteristics would bridge this gap to make efficient use of the current network processing algorithms.

We give an overview of challenges in analyzing small as well as large networks in Figure 1.1. While large networks have constraints such as storage space, computational and measurement costs, small network analysis face issues such as inaccessibility of data and incompatibility of missing entries with existing algorithms. We cater to these problems of processing large networks by proposing a new way of sampling large networks, in which the compact data can be used to reconstruct the original network for conducting experiments without assumptions.

This also sheds light on the flaws of network representation methods. Networks are generally represented as graphs that can be stored in the form of an adjacency matrix to stockpile all the edges in the network. However, with time, due to easy access to the Internet, and the increasing availability of online applications, networks are growing and changing fast, generating a huge amount of data. Consequently there is a need to represent and store networks in a way that preserves

the original network characteristics for uncompromising network experiments while reducing the computational costs and meeting measurement constraints.

1.2 Contributions

The goal of this research is to develop solutions for existing challenges in network topology analysis in the absence of complete network measurements. towards this end, we focus on two areas, namely, a) **efficient compact representation** of completely measured networks and b) accurate **estimation of missing network distances** while preserving original network characteristics. After completion of this work, we will be able to present methods to reconstruct the partially available network data concordant with the original network characteristics. We also propose a novel network representation scheme for compressed and effective modeling as well as storage of network topology. We will be investigating both spatial and social (both undirected and directed) networks for reconstruction and make useful conclusions about the network properties from available data.

We intend to work on the listed areas above by dividing them into following sub-problem areas which eventually build towards our efficient network recovery and modeling goals: a) efficient network modeling, b) deterministic prediction schemes for spatial networks, d) deterministic prediction schemes for social networks, e) using neural networks and pre-training for prediction with ultra-sparse samples, and f) efficient pre-training for neural networks with ultra-sparse samples. We briefly explain these areas and our proposed methods towards achieving a possible solution. These objectives are aligned with our main objective of efficient network representation and recovery.

Here, we summarize our contribution towards solving the aforementioned problem statements.

1. **Efficient network modeling**: (Chapter 4)

   This chapter presents modeling of network data using node-pair distances. Main concepts are introduced along with properties of network data. We also present a novel network sampling scheme for compact and lossless graph representation.

2. **Deterministic prediction schemes for spatial networks**: (Chapter 5)

   We introduce a deterministic method, based on Low-rank Matrix Completion, to reconstruct network topology from partial node-pair distance measurements. We focus on spatial networks that can hold geometric localization. Our main contributions are: (1) leveraging the low-rankness of distance matrices to extract network information from partial measurements to predict missing node-pair distances and (2) topology reconstruction for several 2D and 3D benchmark sensor networks.

3. **Deterministic network reconstruction for social networks**: (Chapter 6)

   This chapter demonstrates the use of Low-rank Matrix Completion to predict missing node-pair distances in undirected as well as directed social networks. The method is tested on real-world social networks such as Facebook, Twitter, Election Blogs, Software modules network, and Emails.

4. **Using pre-trained autoencoders for network reconstruction**: (Chapter 7)

   Here, we introduce and demonstrate a non-deterministic method for predicting missing node-pair distances in social networks. Concept of pre-training is used for transfer learning. We solve the problem of lack of sufficient training data by using artificial data similar to the network we are predicting on.

5. **Designing an improved pre-training method for neural networks**: (Chapter 8)

   This chapter introduces an improved pre-training method. We build an Oracle that dictates ideal pre-training parameters from very few distance samples. We compare the prediction performance of model pre-trained on accurate parameters for model pre-trained with inaccurate parameters and models without pre-training. The model is evaluated by prdicting distances in real-work social networks (Facebook, Email, Train bombing) with only 1% of known distances.

Our research is motivated from the listed challenges in analyzing large networks. Accordingly, we cater to the current issues by designing **efficient sampling techniques** and **robust network**

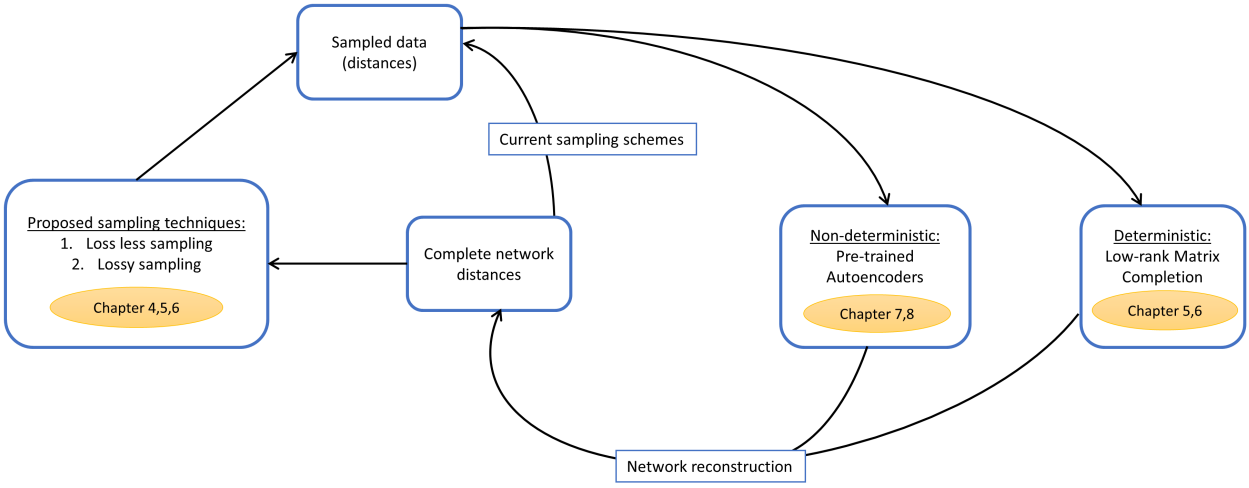


Figure 1.2: Concept map of proposed solutions and interdependence of involved aspects

completion methods. Several other objectives will also be achieved to complement the proposed analysis techniques. Figure 1.2 provides a concept map of proposed solutions and their interconnections.

1.3 Impact and applications

Inability to obtain complete data affects several areas such as estimating original network characteristics, ability to use existing network processing algorithms with partial measurements, and utilizing known network measurements to make decisions about the network. Thus, our imputation methods make an impact in numerous applications. Some of them are listed here, with examples.

Distributed systems:

Systems such as sensor networks or the Internet of Things (IoT) are a collection of several small devices that are equipped with limited computational and energy resources. Storing and processing the complete network distances is not feasible in such cases. The proposed sampling techniques make it possible for each distributed system node to process only partial network distances and still be able to predict the network topology or efficient routing paths.

Recommender systems:

Our methods can be applied to any data that can be represented in the form of graphs. Imagine all the things being sold on an e-commerce platform (ex. Amazon, eBay, Flipkart, etc.). If we were to compute the distance between each object using their features - such as product domain, season this product is sold most in ex. Christmas, rainy season, etc. - we could compute a graph where each node is a product and the links represent similarity or distance between each product. But we might not know all the features for all the products, restraining us from computing all the product distances. By predicting the missing node-pair distances, we could have a complete product graph which can then be used to find the next best product recommendation for the user. Ex. - If a user buys a raincoat, we might suggest her gumboots as gumboots are very close to raincoat in the computed product graph. This can be applied to other systems such as entertainment shows (Netflix), booking websites (Airbnb or flight tickets), etc.

Supporting existing applications:

In social network analysis, a lot of applications such as routing, link prediction, clustering, separation, etc. rely on distances between nodes. In order to make use of these existing applications, we need to first have all the node-pair distances in the graph.

1.4 Outline

Chapter 2 provides a detailed literature review on current challenges in analyzing large networks and related network sampling and representation techniques. Chapter 3 lists our problem statement and objectives. As we have worked on some aspects of theoretical analysis of network data, we present our related findings in Chapter 4. Chapter 5 summarizes our work on the recovery of network topology and other network properties of spatial networks. The extension of the network recovery techniques to social networks is detailed in Chapter 6. Prediction using pre-

trained autoencoders and improved pre-training method are discussed in Chapter 7 and Chapter 8 respectively. The summary of our research proposal and future work are presented in Chapter 9.

Chapter 2

Literature Review

In this chapter, we review the state-of-the-art research and methods which are related to our problem statement or motivate our goals. The literature review focuses on four main areas, namely: a) Challenges in analyzing large networks, b) Network sampling techniques, c) Network reconstruction techniques, and d) machine learning and deep learning techniques for graph processing.

2.1 Challenges in analyzing large networks

Challenges in analyzing large-scale networks have been widely recognized. The main issues while collecting and analyzing a large volume of network data measurements are related to the size of the network, accessibility limitations that prevent complete connectivity information from being observed, missing data due to measurement constraints, and scalability of network processing techniques to large networks [22].

Real-world applications give rise to networks that are often comprised of several components such as users, their connections, user interests, and location to name a few. Graphs are a popular representation for such data because of their ability to represent the different entity and relationship types, including the temporal relationships necessary to represent the dynamics of a data stream. However, fusing such complex interdependent data into a single graph or multiple related graphs and mining them are challenging tasks. Emerging massive data has made such tasks even more challenging [23].

Big Data is characterized by its five V's:

1. Volume: The data we are dealing with is "Big". Size is referred to as volume.

2. Variety: Variety refers to the many types of data that are available. This consists of structured, unstructured, as well as semistructured data types, such as text, audio, and video.

3. Veracity: Veracity refers to the inconsistencies and uncertainty in data collected. Various sources may have different data dimensions, formats, as well as granularity which can make data messy.

4. Velocity: Velocity is the fast rate at which data is received. Some internet-enabled products operate in real-time and thus produce and require data processing in real-time.

5. Value: This refers to the value big data can provide and what organizations can do with correct processing and analysis of big data. The value of data increases with the insights one can gain from it.

While sampling such data (networks) even a small percentage of measurements can make up a very large volume of measurements. On one hand, Big Data holds great promises for discovering subtle population patterns and heterogeneities that are not possible with small-scale data. On the other hand, the massive volume of the sampled data and high dimensionality of Big Data introduces unique computational and statistical challenges, including scalability, storage bottleneck, noise accumulation, spurious correlation, incidental endogeneity, and measurement errors. These issues can lead to wrong statistical inferences and consequently wrong scientific conclusions. High dimensionality combined with a large sample size creates issues such as heavy computational cost and algorithmic instability [24].

The problem of estimating key parameters of large networks (e.g. online social networks) has been actively considered in the context of the World Wide Web [25]. These problems become challenging and interesting with the following possible realistic issues: i) the network is available to us in its entirety and we can only query a node and obtain all its neighbors, ii) these queries are expensive, and thus an algorithm has to make a small number of queries, and iii) it may not be possible to access a uniformly random node in the network.

Visualization and analysis of dynamic networks is a challenge because network measures and layout have to be recalculated for every point in time the network is analyzed. Given the dimension of real-world dynamic networks, a complete recalculation would consume much time and memory.

Two main challenges exist for dynamic network representation. The first challenge is to capture the most relevant features of the dynamic network while eliminating spurious information that does not improve understanding. The second challenge is to represent dynamic graphs efficiently so that the massive volumes of data in these domains can be processed in a reasonable timeframe [26].

Lack of data for research purposes serves as a huge challenge in analyzing large networks. Extremely large graphs are difficult to find outside of a small number of select institutions such as Google and the National Security Agency (NSA). Facebook purports to have a real-world graph on the order of one trillion edges [27], and the National Security Agency purports to have a real-world graph with 70 trillion edges requiring one petabyte of storage [28], the largest publicly available graph has around 128 billion edges [29, 30].

One of the approaches to overcome the lack of data is to evaluate the synthetic networks that can be generated at arbitrary size scales [31, 32]. However, the synthetic data used for training should be a reasonable approximation of the real-world data and should not have any missing entries. We introduce a novel approach to train a prediction model with synthetically generated graphs while keeping the synthetic data characteristics faithful with that of the test data. To the best of our knowledge, this approach has not been used for graphs before.

While large graphs are typically sampled from, filtered, or abstracted to fit within memory requirements, simply accessing and loading these graphs into memory can take hours. Moreover, computationally complex algorithms, such as finding high-order motifs, or simply rerunning algorithms under different experimental conditions, require considerable computational resources [23].

Thus, regardless of how big the data is, it very likely is incomplete and noisy. For example, current maps of the Internet are known to be incomplete and significantly biased [26]. The challenge is to improve the understanding of network characteristics from given measurements. Specifically, given an incomplete and possibly biased view of the network, can we infer network properties (at micro, mezzo, and macro levels) with provable accuracy? And can we design better graph representation techniques or learning algorithms for graph mining tasks?

2.2 Network sampling techniques

Network science research had been confined to small groups because large networks are intractable until network sampling was introduced [33]. Sampling is the selection of a subset (a statistical sample) of measurements from within a statistical population to estimate characteristics of the whole population [34]. Two advantages of sampling are that the cost is lower and data collection is faster than measuring the entire population.

A study done in [10] evaluates the use of an online data collection method to survey early survivors of childhood cancer about their health-related quality of life (HRQOL). It shows that web-based, online data collection methods create opportunities to conduct research globally. However, web-based research requires careful consideration of how the study will be advertised and how data will be collected to ensure high-quality data and validity of the findings.

Research done in [12, 35] employ crawling as a network sampling technique where starting from a random node, the network is crawled to obtain samples. While network crawling or sampling, constraints such as user anonymity, and inaccessibility, may prevent follow up with the participants in the case of missing data. Some networks have public and private nodes which restrict access to user information causing incomplete data. Evaluation of web crawling techniques done in [36] shows that it is a biased and selective network sampling technique.

Snowball sampling, in which existing study subjects recruit future subjects from among their acquaintances, is used to study hidden or specific populations. It is used in [37], for example, to study a group of non-heterosexual women. Snowball sampling [13, 17, 18] is subject to numerous biases such as users with a larger number of friends having more control over the sampling process. This may cause the weaker ties [19] to appear less influential where as the fact can be exactly opposite.

Ego networks consist of a focal node ("ego") and the nodes to whom ego is directly connected (these are called "alters") plus the ties, if any, among the alters [16]. Work in [38] employs ego networks to identify circles on a diverse set of data. However, different participants have different levels of importance depending on the phenomenon to be studied. Ego networks can be over-

simplified and thus, may not support community study or network structure as a whole. Uniform sampling was proposed to obtain an unbiased nodal property distribution as of the original network but it suffers from low efficiency and representativeness of the obtained subgraph [39]. Adaptive uniform sampling (adpUNI) improves on the shortcomings of UNI sampling and presents a very through assessment of their sampling technique for robustness, representativeness, and connectivity of the subgraph [40].

Statistical sampling is useful when sufficient data are not available, the cost of collecting and analyzing every individual observation from a population is too high or time-consuming, or a population is too large to reasonably analyze each observation. However, appropriate sample sizes are contingent, for example, on the variable of interest. Samples that are measured with respect to one variable may not contain enough observations to draw reliable inferences on other variables of interest [41].

Synthetic network models provide possible solutions for two problems: (i) assessing the statistical significance of results in networks and (ii) measuring the performance of new algorithms on extremely large graphs [28]. But there is widespread disagreement about the relevance of the current state of synthetic generators. New models are constantly being proposed to accommodate the latest observed features of real-world networks (see, for instance, [42]).

While these sampling techniques have helped extend network analysis to large networks, they do not capture the graph information to guarantee or allow the reconstruction of the graph. The sampled network is merely supposed to act as a prototype of the original network with deep similarities to it.

Metric dimension [43] of a graph allows us to represent a node as unique distance tuples that capture the node information. However, the link information is not necessarily captured. We propose a fundamentally new compressed representation method that allows us to represent nodes with tuples and also reconstruct the original graph exactly. We intend to employ the current sampling algorithms for optimum data collection so that if and when necessary, the unmeasured data/ facts can be accurately estimated from the sampled data. The proposed technique can be widely

applicable to data that can be represented as graphs and can be employed to bring a fundamental change in the way we process large networks.

Our work is similar in concept, and perhaps even further extends the ideas such as resolvability in graphs with metric dimension and resolution set preserving node information of the given graph [44]. Metric dimension allows us to select a set of nodes that result in distance vectors to the graph nodes which uniquely identifies each node in the network [45]. However, metric dimension does not guarantee a unique and complete reconstruction of all the edges in the graph and hence the sampled distance measurements may not be able to reproduce the original graph.

Virtual coordinate (VC) system is widely used for sampling and network representation and reconstruction in the case of sensor networks for regenerating topologies [46]. As the quality of VCs generated is significantly affected by the selection of anchors, we can say that anchor selection schemes play a vital role in network representation. We intend to design a basis for graphs that can be used as anchors for generating robust VCs that can preserve and reproduce the original network.

2.3 Network reconstruction techniques

While considering the existing network sampling techniques, it is important to note that with incomplete data collection, the hop-distances acquired or the links observed are only partial entries in the complete distance or adjacency matrix of the graph represented by such networks. As shown in [20], missing data can have a significant impact on the ability to estimate structural properties (clustering coefficient, average path length) of a social network. Missing or incomplete data changes the representative network significantly, causing it to deviate from the original network or node feature values, and therefore creates a gap between network data and meaningful network analysis. Estimating the missing entries and completing the available partial data in accordance with the original network characteristics bridges this gap to make efficient use of the current network analysis techniques. Matrix completion is the task of filling in the missing entries of a partially observed matrix [47]. Research in [48] uses matrix completion to investigate the network completion problem, where it is assumed that only a small sample of a network (e.g., a complete or

partially observed subgraph of a social graph) is observed, and to infer the unobserved part of the network. Authors of [48] assume that besides the observed subgraph, side information about the nodes such as the pairwise similarity between them is also provided. In contrast to the original network completion problem where the standard methods such as matrix completion are inapplicable due to the non-uniform sampling of observed links, the authors show that by effectively exploiting the side information, it is possible to accurately predict the unobserved links.

The problem of predicting missing network measurements is cast in the Expectation Maximization (EM) framework where the authors use the observed part of the network to fit a model of network structure, and then estimate the missing part of the network using the model, re-estimate the parameters and so on [49]. Work in [50] uses matrix completion for uniformly sampled data to improve on prior work by [51–53]. This general problem arises in several different settings in information retrieval, social network analysis, and computational biology [54, 55]. Network completion has been approached by other methods as well. Research in [56] presents a global study on uncovering financial network structures from partial data. To infer biochemical interactions from large-scale observations, such as transcriptomics, proteomics, and metabolomics [57] proposes to use a partial correlation analysis. Statistical methods for gene network reconstruction using expression data are reviewed in [58].

The problem of link prediction has also been actively addressed by using machine learning techniques [59]. There are several proximity based supervised machine learning techniques designed to predict missing links in social networks where proximity indicates a similarity score between two given nodes. Authors of [60] present experiments on large co-authorship networks which suggest that information about future interactions can be extracted from network topology alone, and that fairly subtle measures for detecting node proximity can outperform more direct measures. Research in [61] shows empirically, that taking into account the bipartite nature of the graph can enhance substantially the performances of prediction models we learn. Authors of [62] work on efficient and accurate proximity estimation in large social networks with millions of nodes. The paper presents two novel methods, proximity sketch and proximity embedding. They propose

to achieve link prediction in two ways: a) by deriving theoretical model of relationships between connected and non-connected nodes in the network, and b) with machine learning approach while using virtual coordinates as a proximity measure in static networks. Our goal in this research is not to predict links but the missing node-pair distances. Note, that links can be predicted as a byproduct of our prediction schemes.

2.4 Pre-training neural networks for social network analysis

The conventional deep learning methods, such as convolutional neural networks and recurrent neural networks, have mainly focused on the grid-structured inputs of image and audio. Leveraged by representation learning capabilities, deep learning-based techniques can detect structural characteristics of graphs, giving promising results for graph applications. Transfer learning is most effective when the source and target domains bear a high level of structural similarity in their graph representations [63].

Recently, deep learning methods have been proposed to automatically extract structural characteristics from graphs [64]. The localized SCNN model [65], which is a deep learning approach can extract the properties of deformable shapes. The generalized SCNN model [66, 67], borrowed the Fourier transform concept from the signal processing field to apply CNNs in a grid domain to a graph-structured domain. In this model, the convolutional operation was redefined for graphs. A CNN-based model for handling tree structures in the context of programming language processing was introduced in [68].

More recently, the restricted Boltzmann machine [69] was used to learn structural features from graphs in an unsupervised manner for classification [70]. For efficient and scalable semi-supervised classification of graph-structured data, the first-order approximation of spectral graph convolutions was utilized [71].

Graph-based autoencoder framework is used in [72] to perform matrix completion on a recommender system matrix of user-item ratings. The model can make use of side information such as features for users or items which improves the prediction accuracy. As training is solely dependent

on the observed ratings, the loss function only optimizes over the observed ratings. In cold start analysis, where a model is tested for missing ratings, 85% of entries are still known during estimation. We improve on this by starting with data that has from 20% to 99% of entries missing. This provides a huge flexibility of problems we can apply the proposed prediction scheme to. Medium Gaussian SVM along with node proximity features are used in [73] to predict fake accounts in network graphs.

Neural networks have proved to be useful in solving many prediction problems but neural networks often require a large amount of training data to perform well. Pre-training has alleviated this issue with transfer learning, especially when the second task is much more difficult for training [74, 75]. However, relatively little is known about the pre-training behavior related to neural networks used for social network analysis as social networks are difficult to measure or sample completely due to their cost of measurement, privacy, or storage issues. The ever-increasing size of social networks today only adds to this problem.

Our focus or contribution is not towards the generation of artificial graph data. However, it is towards efficient training of the neural networks used in processing graph data. We base our models solely on the observed entries in the matrix to be predicted and do not rely on node features. Having said that, extending our work to employ node features might improve the prediction accuracy. This can also be done by using feature-based models in an ensemble with our existing model.

Chapter 3

Problem Statement

The field of network science has introduced network analysis to extract patterns from graphs that are not in general noticed by humans and to manage network resources over time by gaining insight into how the network is evolving. Network analysis helps us understand the patterns in any network and its organization. A lot of companies (Netflix, Facebook, Google) use network analysis to understand how social factors interact and to help provide a better user experience over a given platform. In fields such as immunology and bio-informatics interactions are recorded, represented, and analyzed as graphs. Many of these applications rely on knowing the network topology. Network topology can be completely captured by the pairwise adjacency information among all pairs, or equivalently the pairwise shortest path lengths. Figure 3.1 gives a brief illustration of how spatial as well as non-spatial networks utilize node-pair distances in real-world. Looking at the wide applications of network science and their dependence on distances, here, we introduce the reader to the motivating factors for our research and give a systematic breakdown of the proposed area of research.

Ideally, one should have the complete network measurements (adjacencies or distances) regarding the network being studied to be able to arrive at meaningful deductions about the original network. However, it is not often possible to capture the entire topology of the graph due to factors such as accessibility constraints, storage limitations, and measurement complexities to process complete large networks due to computing and storage limitations. Thus, various network sampling techniques are used to approximate the original voluminous network data into a compressed form.

Firstly, these sampling techniques focus on creating a representation of the original network which will possess similar network characteristics. Whereas, we focus on sampling the network to be able to reconstruct the original network from the compact sampled data. Secondly, the gap created by missing node-pair distances in the state-of-the-art sampling techniques may affect

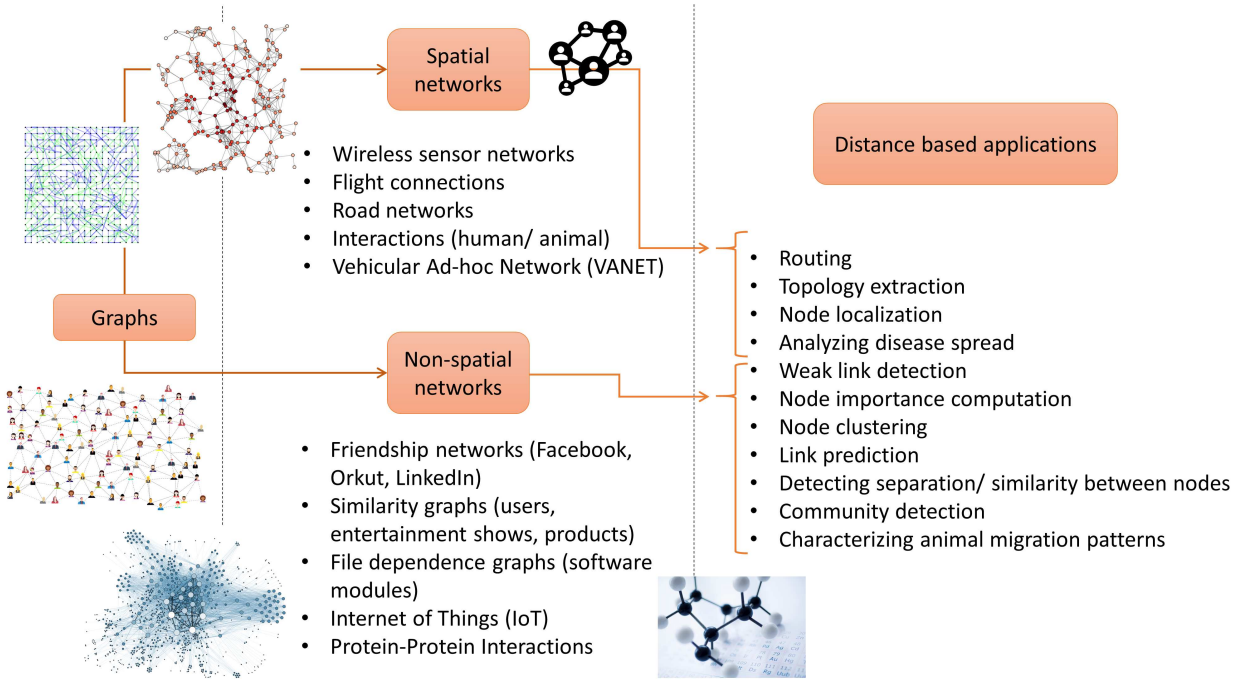


Figure 3.1: Applications in spatial and non-spatial graphs that utilize node-pair distances

our ability to estimate original network features. Partial knowledge of distances influence the effectiveness and accuracy of results, consequently creating the need to efficiently predict missing node-pair distances and coherent but compressed models for network representation.

Both the issues mentioned above: a) accurate **estimation of missing node-pair distances** while preserving original network characteristics and b) efficient **compact representation of large networks**, are critical for efficient analysis of a network at low measurement, computational, and storage costs as well as to support existing network analysis applications. This poses some important research questions to be answered:

- How can we recover the missing entries in a distance matrix from available network measurements?
- How can we retrieve the missing network features from partially sampled node-pair distances?
- How much missing data can we tolerate and still effectively predict the missing entries?

- How can we efficiently sample large real-world networks to preserve its intrinsic features?
- How can we model a graph in a compressed yet lossless manner so as to reconstruct the graph to exactly its original form?
- What features (betweenness centrality, clustering coefficient, etc.) do virtual coordinates of a network preserve about the network?

3.1 Goals

To find techniques for efficient recovery, measurement, and representation of large networks we weave our solution to address listed research questions. We propose to achieve the following goals towards missing data recovery and efficient compressed representation of large graphs.

Goal 1: Develop scalable sampling techniques for networks while preserving the intrinsic network features.

Investigate techniques for sampling real-world networks which can efficiently represent the original network and its features such as clustering coefficient and degree distribution. Sampling distances to a set of anchors in sensor networks for preserving its topological properties works well, however, selecting the best anchors for the given network is NP-complete and we cannot directly map this technique to all networks. The goodness of anchors may depend on factors such as betweenness, node degree, neighbor degree, etc. We propose to design an efficient compression technique for very large networks. This is done by identifying the "defining" nodes and sampling related measurements. These methods need to help reduce the storage cost as well as the computational expenses of large network analysis. The efficiency of our compression techniques is also evaluated with respect to how well the network characteristics are preserved. We also explore concepts beyond metric dimension for graphs, which only allows for unique node labeling, to novel dimensionalities for complete lossless construction of graphs.

Goal 2: Develop efficient algorithms and techniques to retrieve missing information from the available partial data

Design techniques to analyze partial data in the form of distances, links, or node features to study the network topology, patterns, and characteristics. Derive network properties from the available data and relationship among the known entries. Such properties will be used to predict accurate bounds on the missing entries. We use matrix completion algorithms to project approximate values of all the missing entries to provide complete data. Work on novel concepts to make a posteriori conclusions in terms of probability about presence or absence of an unknown link between any two given nodes, based on node features. We achieve this in two scenarios, a) with knowledge of certain constraints on the given network and b) without any constraints.

Goal 3: Compare various techniques for their effectiveness in predicting missing data

Employ and test parametric as well as non-parametric techniques such as neural networks for their effectiveness in predicting missing node pair distances. Evaluate the methods for their missing data tolerance and outline the strengths and weaknesses of each method. Examine neural network prediction models under various conditions to interpret the significance of obtained results. Develop techniques that exploit the known characteristics of the network to partially train the neural network prior to fine tuning with actual measurements. Train neural networks with artificially generated graphs to push the operating missing data percentage of the model as high as possible.

3.2 Outcomes

1. Modeling and estimation of network data for topological properties (Goal 1)

   (a) Network representation

   A novel network sampling technique is presented for unique and complete reconstruction of graphs. The concept of "link dimension" provides the minimum cardinality of

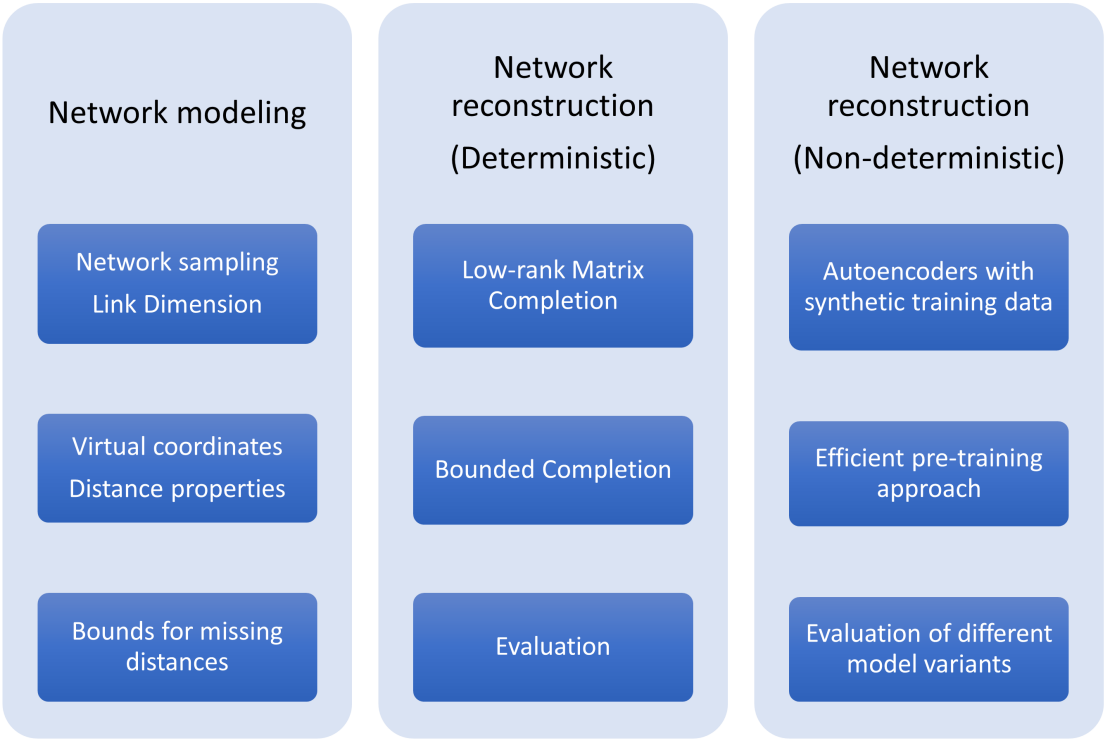


Figure 3.2: Overview of research

the anchor set required to capture the presence and absence of links between any two given nodes. Context-specific alternatives for VCs have been developed.

(b) Virtual Coordinate properties and derivations

The distance measurements in a graph are investigated to derive relationships between various entries in a distance matrix and adjacency matrix.

(c) Reconstruction of graphs

The concept of "link dimension" is extended to reconstruct the original graph, which to a great extent is based on partially available data and/ or known constraints.

2. Extracting information from partial data using deterministic method (Goal 2)

(a) Recovery of missing hop-distance entries using matrix completion

We leverage the low-rank nature of distance matrices of social networks to predict the unknown entries and fill the information gap formed by network sampling techniques.

(b) Utilize bounds for missing distance entries by observing available entries for matrix completion

By using the known data, upper and lower bounds on missing distance entries are computed prior to performing matrix completion. We compare the performances for estimating unknown entries in both cases, i.e., with and without the knowledge of bounds.

(c) Compression analysis for social network representation

Unconventional matrix completion friendly sampling techniques are investigated for networks for efficient capture of intrinsic network features. The technique is analyzed for its compression and performance.

3. Utilize neural networks to reconstruct distance matrix with ultra-sparse samples (Goal 3)

(a) Recovery of missing hop-distance entries using neural networks

We use vanilla autoencoders to learn a compact hidden representation of a network which can be used to predict missing entries in its distance matrix. We leverage the low-rank nature of distance matrices for social networks to predict the unknown entries.

(b) Generate synthetic graph data which is faithful to the target network we are predicting on. We use synthetic data to pre-train the autoencoder as partial distances are not sufficient to efficiently train a neural network.

(c) Design a system to estimate ideal parameters of the synthetic data to be generated for the given partial network we need to predict on.

Thus, we presented the main areas of our research and our proposed approach towards designing these solutions. Figure 3.2 gives a visual overview of our research. The objectives give a systematic breakdown of our goals in terms of tasks completed.

Chapter 4

Modeling and Estimation of Network Data for Topological Properties

In this chapter, we focus on the structural aspects of a network. Thus, when we refer to the term network data, we mean the structural data about the network such as node distances, connectivity information, etc. Network data is typically presented in the form of an adjacency matrix and distance matrix. In this chapter, we observe the relationship between different entries of these matrices to derive network properties. This is important to uncover the hidden information which can be leveraged to make better predictions about missing entries. From the properties derived, accurate upper and lower bounds of unknown entries are estimated and the adjacency matrix can be constructed from the partial data available. Construction of the adjacency matrix from partial data is equivalent to constructing the original graph. The extent to which this can be achieved depends on available entries of the distance matrix and known constraints about the graph. Here, we also propose a novel lossless yet compressed sampling technique for graphs, "Link Dimension".

4.1 Introduction

Algorithms for self-organization and data dissemination among sensor nodes play a crucial role in the performance and life span of large-scale Wireless Sensor Networks (WSNs). Sensor networks have found applications in various fields today. Virtual Coordinate system has been very popularly used for localization of nodes in such networks. VCs, which are functions of hop-based distances from a set of nodes in the network to all the nodes, are not dependent on satellite signals (unlike GPS locations) nor the geographical distribution of the sensor nodes. They only depend on the connectivity of the nodes. Due to low energy and low computational cost requirements, VC systems have increased the ease of deployment of sensor networks in a lot of areas. However, VC systems have faced a common problem in most of the areas, i.e., deducing the location of an event

and structure of the original network. Extracting network topology from the existing information has been one of the ways of knowing about the shape of the network and the location of the sensors. Multi-Dimensional Scaling (MDS) [76], Singular Value Decomposition (SVD) [46], etc. are various methods used to date for extracting a spatial graph of the network from lossy VC matrix. The basic idea is to preserve the connection information in the networks to reconstruct the original graph. The resultant matrix is expected to store not only the distances but also the links between any two nodes in such a way that each node and its links from the original network can be predicted from the matrix itself, in the most compact way possible.

Link mining is a subset of data mining techniques that explicitly consider these links when building predictive or descriptive models of the linked data. Commonly addressed link mining tasks include object ranking, group detection, collective classification, link prediction, and sub-graph discovery [77].

In this article, we look into the intrinsic properties of VCs and predict the original graph using merely the VCs. We call it the "Resurrection method". Though we are presenting this method in reference to sensor networks, it can be applied to any object which can be interpreted as a graph, e.g. chemical compounds, social networks, neural networks, etc.

4.2 Graph representation

A graph is a collection of vertices and edges. Vertices and edges can be represented by several structures, such as social networks, neural networks, chemical compounds, etc. Thus, graph theory finds its applications in a lot of areas.

In networking, graphs can be used to represent a variety of networks using the nodes and connections between them. Nodes can be sensors or computers and connections can be established in a wired or wireless way, literal or virtual as in the cases of LAN and social networks. A graph is represented in terms of matrices that store its characteristic features. A labeled (chemical) graph may be associated with several matrices. Two very important graph-theoretical matrices are the vertex-adjacency matrix (also known as simply the adjacency matrix) and the distance matrix.

Consider a simple undirected connected graph G, defined by $G = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V}$ is the set of nodes of cardinality N and $\mathcal{E}$ is the set of edges (links). G may be represented in terms of its adjacency matrix $\mathbf{A}$, where

$$\mathbf{A} = [a_{ij} | a_{ij} = 1 \text{ if } (i,j) \in \mathcal{E}, \text{ 0 otherwise }]. \tag{4.1}$$

Nodes i and j are said to be adjacent if $a_{ij} = 1$. Alternatively, G can be represented by its distance matrix $\mathbf{H} \in \mathbb{Z}^{N \times N}$, where the ij^{th} element is given by h_{ij}, such that

$$\mathbf{H} = [h_{ij} | h_{ij} = \text{the number of links in the shortest path from node } i \text{ to node } j]. \tag{4.2}$$

Let $\mathcal{M} = \{A_1, ...A_m\}$ be a subset of nodes, such that $\mathcal{M} \subset \mathcal{V}$, with cardinality $m = |\mathcal{M}|$, designated as landmarks. The ordered set of distances to each landmark from all $i \in \mathcal{V}$ forms a *distance vector matrix* $\mathbf{P}_{\mathcal{M}}$ of size $(N \times m)$, where the i^{th} row of $\mathbf{P}_{\mathcal{M}}$ is the distance vector (DV) [78]

$$\mathbf{P}_{\mathcal{M}}(i) = \langle h_{iA_1}, h_{iA_2}, .., h_{iA_m} \rangle; \;\; i = \{1, 2, .., N\}. \tag{4.3}$$

Note that $h_{iA_k} = 0$ for $i = A_k$ and $h_{iA_k} > 0$ for $i \neq A_k$. $\mathbf{P}_{\mathcal{M}}$ consists of a subset of columns of $\mathbf{H}$.

Distance vector based methods are attractive for many communication and social networking operations and applications such as network discovery and verification [79], localization of nodes in a network [80], robot localization problem [43, 81], etc. In the context of networking, graphs are termed as networks, landmarks as anchors, and distance vectors to landmarks for nodes as *virtual coordinates* [77, 82, 83]. Virtual coordinate based techniques are used to overcome, for example, uncertainties of physical distance measurements required for Cartesian coordinates caused by fading or interference of radio signals in wireless sensor networks [84] and in contexts such as social networks where Cartesian coordinates are meaningless [85]. Conceptually, the adjacency matrix (Eq. (7.1)) and the distance matrix (Eq. (8.6)) are equivalent in representing a graph as one can be derived from the other. However, as explained in [79] using examples from communication

networks, it is often realistic to obtain the distance (i.e., h_{ij}) between node pairs in many communication networks, while it is difficult or impossible to obtain information about the presence or absence of specific edges (i.e., a_{ij}) that are far away from the query node. Landmarks, for the distance vector matrix $\mathbf{P}$, are typically chosen randomly or based on heuristics [86], only to ensure the uniqueness of coordinates of nodes without regard to the suitability of resulting DVs for the reconstruction of the graph or to capture the topology. Topology or layout information is recovered from these distance vectors using approximations such as low-rank matrix completion [87] or by exploiting the statistical characteristics observed in the class of networks [88]. There is no formal basis to relate the landmark selection and the corresponding distance vectors to the ability to recover the network topology, a problem that we address in this paper.

With a sufficient number of landmarks, the rows of $\mathbf{P}_{\mathcal{M}}$ can uniquely label each node in G. A set of such landmarks is called a *resolution set*, $\mathcal{R}$, and the minimum possible cardinality of such a set is called the *metric dimension*, $\beta(G)$, of G. Let $\tilde{\mathcal{R}}$ be such a minimum resolution set; thus $\beta(G) = |\tilde{\mathcal{R}}|$. The concept of metric dimension was presented in [43,89] and [44,46,90–92] among others extended the results for different families of graphs.

Several variations of metric dimension and related generators have been introduced in the literature, for instance, resolving dominating sets [93, 94], independent resolving sets [95], local metric sets [96], k-metric generators [97], and resolving partitions [98]. Among these, edge metric dimension [99] and strong metric dimension [100], [92] are relevant to the concept introduced in this paper.

Edge metric dimension for a graph G, $edim(G)$, helps to uniquely distinguish each edge in a graph using the distances of each edge from nodes in edge metric generator, $\mathcal{S}$, where the distance between the node $v, v \in V$, and the edge $e, e = (u, w) \in E$, is defined as $d_G(e, v) = min\{d_G(u, v), d_G(w, v)\}$. Thus, the distance vectors corresponding to $edim(G)$, $\mathbf{P}_{\mathcal{S}}$, informs about each of the edges in G [99].

Strong metric dimension, *sdim*(G), is the minimum cardinality of a strong resolution set [92, 100]. Node w strongly resolves two nodes u and v if u belongs to a shortest $v - w$ path or if v

belongs to a shortest $u - w$ path. The node set $\mathcal{S}$ is a strong resolving set if every two distinct nodes of G are strongly resolved by some vertex in $\mathcal{S}$.

Although the existing approaches allow the representation of each node or each edge of G with unique virtual coordinates, in general, multiple graphs satisfy the same set of coordinates, and therefore the construction of the original graph from the distance vectors is not possible. We address the construction of a graph from distance vectors to a small set of landmarks, an especially important operation for large-scale networks as it dramatically reduces the complexity of network measurement, topology extraction, and other network analytics [85, 88].

Section 4.1 presents the introduction and motivation for better graph representation schemes. The data used for network representation and its properties are explained in Section 4.2 along with the necessary terms in network connectivity. Section 4.3 talks about challenges encountered by metric dimension for reconstructing a graph. Section 4.4 presents a graph reconstruction algorithm. The novel concept of "Link Dimension" for lossless graph compression and exact reconstruction is introduced. Subsequently, various properties and bounds on Link Dimension for different graph families are derived in Section 4.5. Section 4.7 talks about ambiguity while reconstructing graphs from their resolution sets and how Link Dimension solves this problem. Spectral moments and their correlation with graph dimensions are studied for all possible graphs for a given number of nodes in Section 4.8. Section 4.9 summarises the main contributions of this work.

4.3 Metric dimension and graph reconstruction

Let $\mathbf{P}_{\tilde{\mathcal{R}}}$ be the $(N \times \beta(G))$ matrix of unique distance vectors for $\tilde{\mathcal{R}}$. Though the distances from nodes in $\tilde{\mathcal{R}}$ to other nodes allow unique representation of each node, we show that $\mathbf{P}_{\tilde{\mathcal{R}}}$ does not guarantee the ability to reconstruct the original graph which a complete $\mathbf{H}$ or $\mathbf{A}$ does.

Theorem 1. *The set of distance vectors, $\boldsymbol{P}_{\tilde{\mathcal{R}}}$, corresponding to a minimum resolution set, $\tilde{\mathcal{R}}$, of graph G, does not guarantee exact recovery of G.*

Proof. Consider the two graphs shown in Figures 4.2 $[a]$ and 4.2 $[b]$. Note that each graph is compatible with the unique DV set $\mathbf{P}_{\tilde{\mathcal{R}}} = \{\langle 0, 1\rangle, \langle 1, 0\rangle, \langle 2, 1\rangle, \langle 2, 2\rangle, \langle 1, 2\rangle\}$ corresponding to the

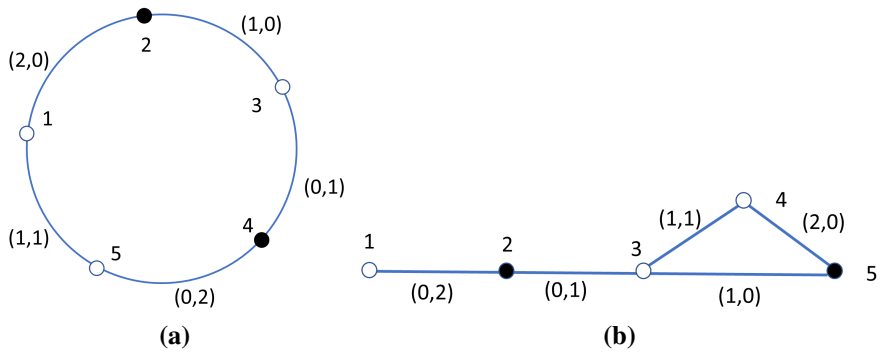


Figure 4.1: Edge metric dimension for graphs. Graphs in (a) and (b) have the same distance vectors $\mathbf{P}_{\mathcal{S}} = \{\langle 0,2\rangle, \langle 0,1\rangle, \langle 1,0\rangle, \langle 2,0\rangle, \langle 1,1\rangle\}$ generated from their edge metric generators (dark nodes). Each edge is shown with its distance vector from nodes in their edge metric generator set.

minimum resolution set of nodes $\tilde{\mathcal{R}} = \{1,2\}$. The presence or absence of edge (5,3) does not change $\mathbf{P}_{\tilde{\mathcal{R}}}$ and conversely, the presence or absence of this edge cannot be ascertained from the distance vectors in $\mathbf{P}_{\tilde{\mathcal{R}}}$. □

Similarly, $edim(G)$ for graph in Figure 4.1 $[a]$ is 2 corresponding to the edge metric generator $\mathcal{S} = \{2,4\}$ and thus $\mathbf{P}_{\mathcal{S}} = \{\langle 0,2\rangle, \langle 0,1\rangle, \langle 1,0\rangle, \langle 2,0\rangle, \langle 1,1\rangle\}$. For the graph in Figure 4.1 $[b]$, $edim(G') = 2$ corresponding to $\mathcal{S}' = \{2,5\}$ but the edges can be represented with the same $\mathbf{P}_{\mathcal{S}}$. Also, $\mathbf{P}_{\mathcal{S}}$ does not capture node information. Thus, $\mathbf{P}_{\mathcal{S}}$ does not allow exact reconstruction of G. The question whether there exists an edge between any given pair of nodes cannot always be answered using distance vectors corresponding to edge metric dimension or metric dimension of a graph.

Having a unique graph G corresponding to $\mathbf{P}_{\mathcal{R}}$ of a resolution set $\mathcal{R}$ is desirable as $\mathbf{P}_{\mathcal{R}}$ then provides an alternative compact representation of G. However, there is no formal foundation for the number of distance measurements needed to construct a graph for problems such as topology extraction [85, 88]. The difficulty arises due to edges such as (5,3). Next, we define terms *invisibility* and *ambiguity* of such edges from two perspectives, that of selecting landmarks (i.e., $\mathcal{R}$) for a graph, and of constructing G from $\mathbf{P}_{\mathcal{R}}$ respectively, and illustrate them in Figure 4.2.

Definition 1. *Invisible edge: An edge (i, j) in a graph G is said to be invisible with respect to a resolution set $\mathcal{R}$ when the removal or the addition of edge (i, j) from G does not affect any value in $\boldsymbol{P}_{\mathcal{R}}$.*

Definition 2. *Ambiguous edge: An edge (i, j) is ambiguous with respect to a distance vector set $\boldsymbol{P}_{\mathcal{R}}$, if $\exists$ two graphs G_1 and G_2 that satisfy $\boldsymbol{P}_{\mathcal{R}}$, where (i, j) is present in one and absent in the other.*

Thus, a set of invisible (ambiguous) edges for G with $\mathcal{E}$ as the set of edges, can be formalized in Eq. (4.4).

$$\mathcal{I} = \left\{ (i,j) \;\middle|\; \begin{array}{l} (i,j) \in \mathcal{E} \textit{ and removal of } (i,j) \textit{ from } G \textit{does} \\ \textit{not change the distance vectors of } i \textit{ and } j \\ Or \\ (i,j) \notin \mathcal{E} \textit{ and } \delta_{ij} = 1 \end{array} \right\} \tag{4.4}$$

Lemma 1. *No edge connected to a landmark node is invisible (or ambiguous).*

Proof. Landmark $A_k \in \mathcal{M}$ is the only node with the corresponding coordinate equal to zero, i.e., $h_{iA_k} = 0 \iff i = A_k$. Furthermore, $h_{iA_k} = 1$ if and only if there is an edge between i and A_k. Thus, no edge connected to an anchor is invisible. □

To characterize the difference between distance vectors $\mathbf{P}_{\mathcal{R}}(i)$ and $\mathbf{P}_{\mathcal{R}}(j)$ for a pair of nodes i, j, we define

$$\delta_{ij} = \underset{k}{\textit{argmax}}\{\delta_{ij}^k\} \; \forall \; k \mid A_k \in \mathcal{R}, \textit{ where, } \delta_{ij}^k = |h_{iA_k} - h_{jA_k}|. \tag{4.5}$$

Lemma 2. *An edge (i, j), $i, j \in \mathcal{V}$, is invisible with respect to $\mathcal{R}$ only if $i \notin \mathcal{R}$, $j \notin \mathcal{R}$, and $\delta_{ij} = 1$. Conversely, this is also a necessary condition for an edge (i, j) to be ambiguous w.r.t. a given $\boldsymbol{P}_{\mathcal{R}}$.*

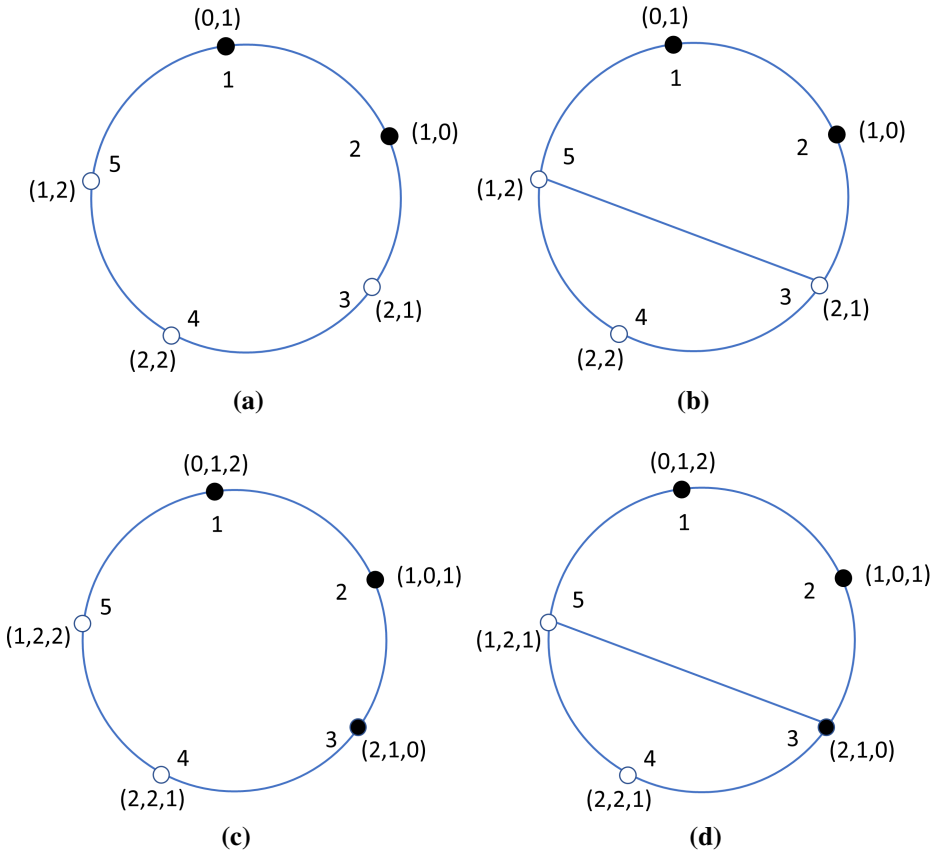


Figure 4.2: Ambiguous and invisible edges: Graphs in (a) and (b) both have $\tilde{\mathcal{R}} = \{1, 2\}$ and $\mathbf{P}_{\tilde{\mathcal{R}}} = \{\langle 0, 1\rangle, \langle 1, 0\rangle, \langle 2, 1\rangle, \langle 2, 2\rangle, \langle 1, 2\rangle\}$. Thus, edge $(3, 5)$ is invisible w.r.t. $\tilde{\mathcal{R}}$, and w.r.t. $\mathbf{P}_{\tilde{\mathcal{R}}}$ edge $(3, 5)$ is ambiguous. Distance vectors of the two graphs are shown in (c) and (d) respectively, when node 3 is added as a landmark, i.e., $\mathcal{C} = \{1, 2, 3\}$. Edge $(3, 5)$ is no longer invisible w.r.t. $\mathcal{C}$, and given corresponding $\mathbf{P}_{\mathcal{C}}$ each graph can be exactly constructed.

Proof. Nodes have unique distance vectors in the case of a resolution set, and thus $\delta_{ij} \neq 0$ for $i \neq j$. From Lemma 1, for an edge to be invisible, $i, j \notin \mathcal{R}$. Consider edge (i, j), $i, j \notin \mathcal{R}$, and a landmark A_k such that $\delta_{ij}^k = |h_{iA_k} - h_{jA_k}| > 1$. Without loss of generality, let $h_{iA_k} \geq h_{jA_k}$. If there is an edge (i, j), then there is a shorter path from A_k to i consisting of the shortest path from A_k to j followed by the edge (i, j) and thus, $h_{iA_k} = h_{jA_k} + 1$ which contradicts the assumption. Thus, given $\delta_{ij}^k > 1$, there can be no edge, invisible or otherwise, between nodes i and j. □

Definition 3. *Feasible links: An edge is feasible between nodes i and j if $\delta_{ij}^k \not> 1 \; \forall \; k$, i.e., $\delta_{ij} \not> 1$.*

Thus, the set of feasible edges is given by

$$\mathcal{F} = \left\{ (i,j) \;\middle|\; (i,j)|\delta_{ij} = 1 \quad \right\} \tag{4.6}$$

4.4 Construction set and link dimension

Next, we introduce the novel concepts of *construction set* and *link dimension* to facilitate the exact representation and reconstruction of a graph via a set of distance vectors.

Definition 4. *Construction Set and Link Dimension: A "construction set", $\mathcal{C}$, is defined as a set of landmarks that allows the exact and unambiguous construction of G from the formed distance vectors, $\boldsymbol{P}_{\mathcal{C}}$, by imposing the condition that an edge exists between nodes i and j if and ony if $\delta_{ij} = 1$. Such a construction set with minimum cardinality is a minimum construction set, $\tilde{\mathcal{C}}$, and its cardinality is called the "link dimension," $\xi(G)$ of G.*

By definition, there exists a unique graph G associated with the distance vectors $\mathbf{P}_{\mathcal{C}}$ of a given construction set $\mathcal{C}$. For example, $\tilde{\mathcal{C}} = \{1, 2, 3\}$ is a minimum construction set for each of the graphs shown in Figure 4.2 $[c]$ and 4.2 $[d]$, and the corresponding $\mathbf{P}_{\tilde{\mathcal{C}}}$s uniquely identify the respective graphs. In contrast, $\tilde{\mathcal{R}} = \{1, 2\}$ for graphs in Figure 4.2 $[a]$ and 4.2 $[b]$ result in the same $\mathbf{P}_{\mathcal{R}}$. However, there may be multiple minimum construction sets associated with G.

Theorem 2. *The necessary and sufficient conditions for a set of landmarks $\mathcal{M}$ to be a construction set, $\mathcal{C}$, are a) $\mathcal{M}$ is a resolution set, and b) for each non-adjacent node pair i, j in G, $\exists$ at least one landmark A_k, such that $\delta_{ij}^k > 1$ (i.e., $\delta_{ij} > 1$).*

Proof. A construction set $\mathcal{C}$ cannot have any two nodes i and j with identical distance vectors, as it would result in an ambiguous edge (i, j). Therefore no two distance vectors are identical and $\mathcal{C}$ is a resolution set. Furthermore, if $\delta_{ij} = 1$ for a pair of non-adjacent nodes i, j, it is possible to add the edge (i, j) without changing the distance vectors resulting in another graph satisfying the distance vectors. Therefore, a construction set has to ensure that $\delta_{ij} > 1$ for any non-adjacent pair of nodes. Therefore conditions (a) and (b) are necessary for a given $\mathcal{M}$ to be $\mathcal{C}$. Now consider the

sufficient conditions. If M is a resolution set, $\delta_{ij} \geq 1$ for any pair of nodes. When condition (b) is satisfied, $\delta_{ij} = 1$ only for adjacent pairs of nodes, and $\delta_{ij} = 1$ only for non-adjacent i and j. Therefore the edges are uniquely defined and the corresponding graph is unique. □

Corollary 0.1. *The link dimension $\xi(G)$ and the metric dimension $\beta(G)$ of a graph G are related by,*

$$\xi(G) \geq \beta(G). \tag{4.7}$$

Proof. A minimum construction set $\tilde{\mathcal{C}}$ is a resolution set, while a minimum resolution set does not necessarily have a sufficient number of landmarks to ensure exact reconstruction of G (e.g., see Figure. 4.2 $[a]$, 4.2 $[b]$). A resolution set is not a construction set when there are edges that are invisible to its landmarks. □

4.4.1 Derivation of a construction set $\mathcal{C}$ from $\mathcal{R}$

Given a resolution set $\mathcal{R}$ for a graph G, a construction set can be obtained as follows:

1. Identify the subset of feasible edges, $\mathcal{F}'$ such that for each edge (i, j) $\delta_{ij} = 1$, but the edge does not exist in graph G:

$$\mathcal{F}' = \left\{ (i,j) \,\middle|\, (i,j) \notin \mathcal{E} \text{ and } \delta_{ij} = 1 \right\} \tag{4.8}$$

2. Select additional landmarks to resolve the edges in $\mathcal{F}'$.

A simple but not necessarily an optimal choice for resolving an edge (i, j) in $\mathcal{F}'$ is to select either i or j as a landmark. Resolving multiple edges using one landmark will produce a more compact solution for $\mathcal{C}$. Algorithm 1 provides a greedy method for obtaining $\mathcal{C}$ from $\mathcal{R}$. The resultant set of landmarks, $\mathcal{C}$, is a construction set for graph G. E.g. in Figure 4.2, $\mathcal{C} = \{1, 2, 3\}$; however, $\mathcal{C}' = \{3, 4, 5\}$ is also a valid construction set. Starting with $\mathcal{R} = \{1, 3\}$ would lead to $\mathcal{C}$ and starting with $\mathcal{R}' = \{3, 5\}$ would give us $\mathcal{C}'$. This greedy approach results in a minimal $\mathcal{C}$, i.e., removal of any node from this $\mathcal{C}$ set may cause ambiguous edge(s). Thus, we can treat $|\mathcal{C}|$ as an upper bound on $\xi(G)$. This method may or my not yield the minimum construction set, $\tilde{\mathcal{C}}$.

Table 4.1: Notation used in the text and algorithm

Notation	Description
$\mathcal{N}$	Set of nodes
$N = \lvert\mathbb{N}\rvert$	Number of network nodes
$n_i \in \mathcal{N}$	i^th node
$\mathcal{A} \subset \mathcal{N}$	Set of anchor nodes
$\mathbf{M} = \lvert\mathcal{A}\rvert (M \ll N)$	Number of anchors
$A_i \in \mathcal{A}, i = 1 : M$	i^th anchor
h_{ij}	Minimum hop-distance between nodes n_i and n_j
$\mathbf{P}_{(i)} = [h_{iA_1}, h_{iA_2}, \ldots, h_{iA_M}]$	VCs of node n_i
$\mathbf{P}_{N\times M} = [h_{iA_j}]_{ij} s.t. i = 1 : N; j = 1 : M$	VC set for entire network
$\mathbf{Q}_{R\times M}; R \ll N$	VCs of a subset of nodes
K	Set of one hop neighbor nodes

Algorithm 1 Derivation of $\mathcal{C}$ from $\mathcal{R}$: A greedy approach.

INPUT: Resolution set $\mathcal{R}$ and $\mathbf{A}$ for a graph G
OUTPUT: Construction set $\mathcal{C}$

1: **procedure** DERIVE $\mathcal{C}$ FROM $\mathcal{R}$
2: $\mathcal{C} = \mathcal{R}$
3: Evaluate $\mathbf{P}_{\mathcal{C}}$ using $\mathcal{C}$ and $\mathbf{A}$
4: Identify the subset of feasible edges that does not exist in the graph, $\mathcal{F}'$ using $\mathbf{P}_{\mathcal{C}}$ and $\mathbf{A}$:

$$\mathcal{F}' = \left\{ (i,j) \;\middle|\; \delta_{ij} = 1 \textit{ and } a_{ij} = 0 \right\} \forall\, i, j \in \mathcal{V} \tag{4.9}$$

5: Select the node that appears in the highest number of edges in $\mathcal{F}'$ and add it to $\mathcal{C}$
6: Repeat steps 3, 4, and 5 until there are no edges in $\mathcal{F}'$
7: **end procedure**

4.5 Properties and bounds

Consider a graph with diameter d, in which case $h_{ij} \leq d \; \forall \; i, j \; \in \; \mathcal{V}$ and therefore, no element of a DV for a non-landmark node can hold values outside 1 to d. The m landmarks themselves have unique DVs with one of the elements equal to zero. Thus, m landmarks yield at most $(d^m + m)$ unique coordinates. Therefore, metric dimension $(\beta(G) = m)$ is bounded by [101]:

$$(m + d^m \geq N) \tag{4.10}$$

Above bound deals with N, the number of nodes to be resolved. In contrast, a construction set has to resolve each of the possible links, i.e., be able to distinguish among each of the L edges present as well as unambiguously exclude edges among non-adjacent nodes. To help obtain bounds for the link dimension, we consider the relationship between DVs for an edge to be feasible. Thus, conceptually, in addition to the bound in Eq. (4.10), link dimension, $\xi(G) = m$, must also satisfy

$$\begin{aligned} \{\textit{Number of feasible links for a coordinate set with } (d^m + m) \textit{ unique vectors}\} \geq \\ \{\textit{Actual number of links in the graph}\} \end{aligned} \tag{4.11}$$

Next, we derive bounds for link dimension, $\xi(G) = m$, under two different constraints:

Case I: The number of links (edges) L of G is known

Link (i, j) is feasible only if $\delta_{ij} \not> 1$. Thus, with respect to edge (i, j), each h_{jA_k}, $k = \{1, 2.., m\}$ is limited to three values from $\{h_{iA_k} - 1, h_{iA_k}, h_{iA_k} + 1\}$. Hence, a given node can have $(3^m - 1)$ possible combinations of adjacent distance vectors corresponding to neighbors or links. For a land-

mark node (A_k) however, the only possible distances for an adjacent node i is $h_{iA_k} = 1$. Thus, a landmark can have only up to $3^{(m-1)}$ links. Thus, Eq. (4.11) can be restated as

$$(3^m - 1) \times (N - m) + (3^{m-1})(m) \geq (L \times 2) \tag{4.12}$$

Case II: Maximum node degree (n_{dmax}) and number of links (L) of G are known

Eq. (4.10) and n_{dmax} yield the bound in Eq. (4.13) for a non-landmark node, while the number of links possible for a landmark imposes the bound in Eq. (4.14), both of which need to be satisfied:

$$(m + d^m) \times \frac{n_{dmax}}{2} \geq L \tag{4.13}$$

$$(3^{(m-1)}) \geq n_{dmax} \tag{4.14}$$

Lemma 3 gives a bound for link dimension when the set of VCs of all nodes in a resolution, $\mathbf{P}_{\mathcal{R}}$, and the miniimun node degree, (n_{dmin}) of graph are given.

Lemma 3. *Given a $\boldsymbol{P}_{\mathcal{R}}$ matrix for a resolution set and minimum node degree (n_{dmin}), of G, an upper bound for link dimension is given by ($|\mathcal{R}| + \epsilon$), where ϵ, the number nodes involved in possible ambiguous edges, is given by, $\sum_{i=1}^{N}\{|\{\mathcal{I}_i\}| - n_{dmin}\}$ where $\mathcal{I}_i = \{j \mid \delta_{ij} = 1\}, \forall\, i, j \in \mathcal{V}$.*

Proof. An ambiguous edge is not connected to a landmark as all the edges associated with a landmark are resolved. From Theorem 2, for $\mathcal{R}$ to be $\mathcal{C}$, all the ambiguous edges need to be resolved. However, an ambiguous link can be resolved by adding one of the nodes associated with that link, to $\mathcal{R}$. While Eq. (4.15) gives the probable links for a node, in Eq. (4.16) we get the upper bound on number of nodes associated with ambiguous links, ϵ, in G by ignoring the minimum links each node has, i.e., n_{dmin}.

$$\mathcal{I}_i = \{j \mid \delta_{ij} = 1\}, \forall\, i, j \in \mathcal{V} \tag{4.15}$$

$$\epsilon = \sum_{i=1}^{N} \{|\{\mathcal{I}_i\}| - n_{dmin}\} \tag{4.16}$$

□

Note, that Eq. (4.16) gives an exact value of ϵ for G where node degree is constant. The set of nodes involved in total ambiguous links for given node i can be derived without reconstructing the graph. The set of ambiguous link in G is given by Eq. (4.15). Thus, for the given $\mathbf{P}_{\mathcal{R}}$, we can get ϵ by Eq. (4.16).

As link dimension is newly introduced here, bounds on link dimension for various graph families are studied. Link dimension bounds for graph families such as linear graphs, complete graphs, finite cyclic graphs, star graphs, 2D grid graph, and wheel graphs are derived here.

Proposition 1. *Linear graph, P_N, is the only graph with $\xi(G) = 1$.*

Proof. If $\mathcal{C}$ has only one landmark, and G has N nodes, the only possible set of DVs is $\{\langle 0\rangle, \langle 1\rangle, ..\langle N-1\rangle\}$. The landmark has the coordinate 0, and it is connected to exactly one node (which has coordinate 1), which in turn is also connected to the node with coordinate 2, etc. This is the line graph, with the anchor placed at one end. □

Proposition 2. *For a complete graph with N nodes, link dimension $\xi(G) = N - 1$.*

Proof. If two of the nodes are not in $\mathcal{C}$, both of them have identical distance vectors, consisting of all 1's. With $N - 1$ nodes in $\mathcal{C}$, each edge is connected to a landmark node and thus all the edges are resolved. □

Proposition 3. *For a finite cyclic graph C_N of $N > 6$ nodes, the link dimension $\xi(G) = 2$.*

Proof. As hop-distance increases radially with distance from the landmark on each side, selecting a single node as a landmark, creates identical distance vector node pairs on opposite sides of the circular graph creating invisible edges as shown in Figure 4.3$[a]$. Square nodes, white round nodes, and triangular nodes form invisible edges between themselves. Selecting two nodes that are one hop apart solves this issue not only by assigning unique DV to each node but also by creating $\delta > 1$ for non-existing edges as shown in Figure 4.3$[b]$. □

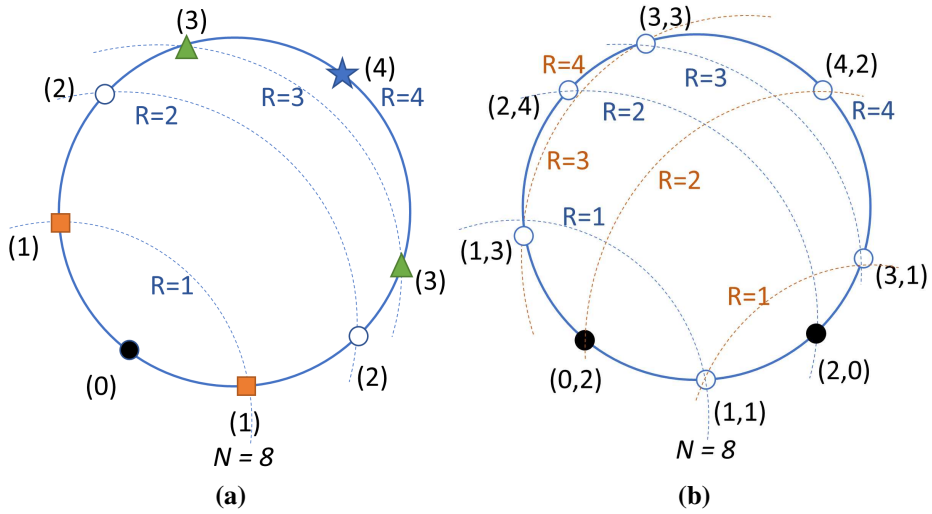


Figure 4.3: Link dimension of Cyclic graphs: (a) Single node is selected as anchor (dark round node). Multiple node pairs (square nodes, white round nodes, triangle nodes) create $\delta_{ij} = 1$ as the radial distance does not resolve each edge as shown. The hop-distance radii are shown with dashed lines. Thus all the dges are resolved. (b) Two non-adjacent nodes are selected as anchors (dark round nodes). The radial distances create $\delta_{ij} > 1$ for non-adjacent nodes, thus, resolving each edge.

For C_N, where $N \leq 6$, $\xi(C_3) = \xi(C_4)$ = 2 and $\xi(C_5) = \xi(C_6) = 3$.

Proposition 4. *For a star graph S_N with N nodes, the link dimension $\xi(G) = N - 2$.*

Proof. The selection of $N-2$ leaf nodes as landmarks creates unique distance vectors. A landmark node has a DV of the form $[2, 2, ..0, ..2]$ with exactly one '0' in it, and therefore it is adjacent to only the center node. The center node is connected to each landmark and is thus resolved with distance vector $[1, 1, .., 1]$ and the non-landmark leaf node has the distance vector $[2, 2, .., 2]$ which is only adjacent to the center node. □

Proposition 5. *For a 2D grid graph, the link dimension is 2, irrespective of the number of nodes N.*

Proof. Here, by 2D grid graph, we mean a rectangular grid of nodes where all perpendicular edges exist [102]. Selecting a single node as a landmark anywhere in a 2D grid graph will cause at least two identical DVs, as shown in Figure 4.4$[a]$. It also creates multiple invisible edges. For instance, the shaded area in Figure 4.4$[a]$ contains invisible edges area associated with diagonal

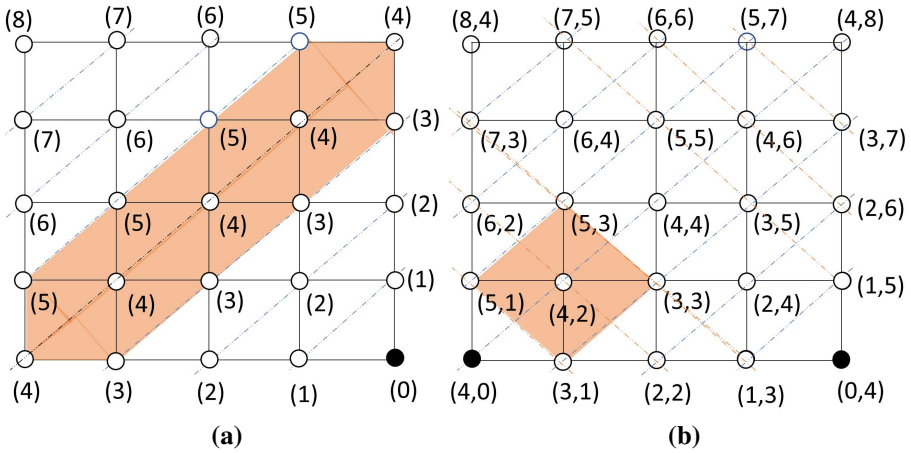


Figure 4.4: Link dimension in a Lattice 2D graphs: (a) A single node is selected as the anchor (dark round node). The shaded area represents nodes that diagonal nodes form invisible edges with. (b) Two corner nodes are selected as anchors (dark round nodes). The shaded area of invisible edges does not include any non-adjacent node. The radial distances create $\delta_{ij} > 1$ for non-adjacent nodes, thus, resolving each edge.

nodes with DV $= [4]$. If any two adjacent corner nodes are selected as anchors, their hop-distances will propagate diagonally to them through the grid. Thus, each equidistant line will cross alternate equidistant lines from other anchors as shown in Figure 4.4$[b]$. This will cause non-adjacent nodes to have $\delta \geq 2$, thus shrinking the invisible edge area to include only adjacent nodes. □

Proposition 6. *For a wheel graph, with number of nodes $N > 5$, the link dimension is $N - 3$.*

Proof. A wheel graph, $\mathcal{W}_N$, consists of two types of nodes, a center node and the rim nodes (nodes on the circumference of the wheel) which are located on the periphery of the wheel. Center node is not a good choice as a landmark as it resolves only itself due to its equidistant placement from all the other nodes. Thus, all landmarks in a minimum construction set have to be selected among the rim nodes.

Consider $N - 4$ landmarks on the rim, i.e., three nodes on the rim are not landmarks, and at least two of those nodes, i and j, are non-adjacent. However, in a wheel graph, a node is either one hop or two hops from any given landmark. Thus, $\delta_{ij} = 0$, indicating that nodes i and j are adjacent. This can be resolved by selecting $N - 3$ landmarks among the rim nodes.

Hence, for a $\mathcal{W}_N$ with $N > 5$, the link dimension is $N - 3$ while the landmarks are adjacent rim nodes. □

To put things in context, if, as has been observed in [92, 103], a strong resolution set can uniquely determine a graph, then $\mathcal{S}$ is a construction set. The question then is whether it is always a minimum construction set. Link dimension provides us with the minimum vertex set for graph reconstruction. So, for example, for the cycle C_N of order N the strong dimension $sdim(C_N) = N/2$ [104], link dimension for C_N for $N > 5$ is 2. Cyclic graph $C_N, N > 6$ shows it is not. According to Proposition 3, $\xi(C_N) = 2$, which is less than $sdim(C_N) = N/2$ [105]. Therefore,

$$sdim(G) \geq \xi(G). \tag{4.17}$$

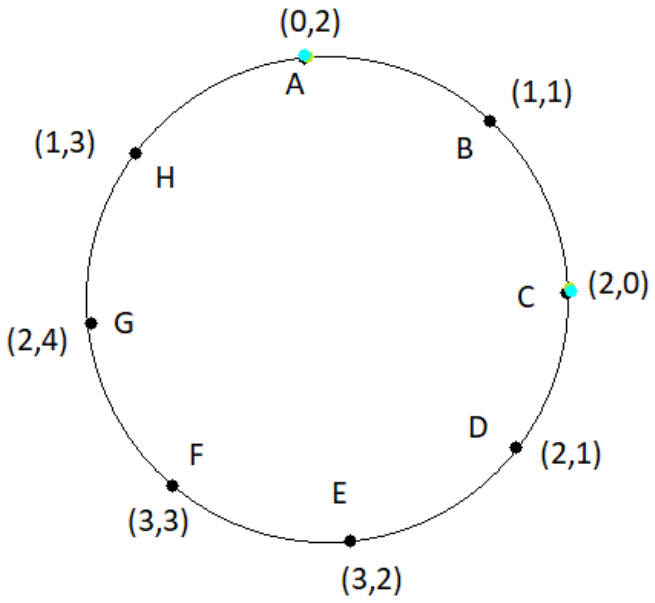


Figure 4.5: An orientation of nodes A_k, i and j without edge (i, j)

This can be easily explained by using a cyclic graph with eight nodes, C_8 (see Fig. 4.5). If we label the nodes as $\{1, 2, 3, 4, 5, 6, 7, 8\}$, $\mathcal{C} = \{1, 3\}$ is sufficient as the construction set for C_8, however, node pair $(2 - 7)$ is not strongly resolved. Thus, $\{1, 3\}$ is not a strong resolution set for C_8.

4.6 Graphability of a distance vector set

Prior work has addressed the distance vectors corresponding to a graph and their relation to the graph. However, a distance vector is merely a non-negative integer vector. Now, we address the question of whether there exists a graph G of N nodes and m landmarks with DV of node i corresponding to the i^{th} row in a given non-negative integer matrix $\mathbf{P}_{\mathcal{M}}$, $\forall\ i$. We approach this by first defining the graphability and then identifying the necessary and sufficient conditions for a given $\mathbf{P}_{\mathcal{M}}$ to correspond to that of a resolution set, $\mathbf{P}_{\mathcal{R}}$, of graph G.

Definition 5. *Graphability: A given set of distance vectors $\boldsymbol{P}_{\mathcal{M}}$ is said to be 'graphable' if there exists a connected graph G corresponding to $\boldsymbol{P}_{\mathcal{M}}$.*

Definition 6. *Reachability of a node from a landmark A_k: A node $i \in \mathcal{V}$ is said to be 'reachable' from a landmark A_k if $\exists$ a node j s.t. $\delta_{ij} = 1$, ($h_{jA_k} = h_{iA_k} - 1$), and node j is reachable from A_k.*

Each node receives its distance coordinate for a particular landmark as the shortest hop-distance from that landmark, i.e., each shortest path passes through a node that is one hop closer to the corresponding landmark. Thus, if a node i has coordinate $(h_{iA_k} > 0)$ with respect to a landmark A_k, $\exists$ a node j in the neighborhood of node i with corresponding coordinate $(h_{iA_k} - 1)$, where neighborhood is defined by all nodes j, such that $\delta_{ij} = 1$. Reachability of node i from a landmark A_k, therefore, guarantees the existence of nodes forming the shortest path of length h_{iA_k} to A_k. In a connected graph, each node is reachable from all the anchors and each anchor, A_k, is reachable from itself.

Theorem 3. *A given matrix of non-negative integer vectors $\boldsymbol{P}_{\mathcal{M}}$ corresponds to a resolution set of a graph with the i^{th} row being the DV of node i, iff the following four conditions are satisfied:*

1. *Each distance vector is unique.*
2. *Each row in $\boldsymbol{P}_{\mathcal{M}}$ has at most one element as '0', and each column in $\boldsymbol{P}_{\mathcal{M}}$ has exactly one element that is '0'.*

3. *Each node $i \in \mathcal{V}$ is reachable from every landmark $A_k \in \mathcal{M}$.*

4. *Each node satisfies triangle inequality with any pair of landmarks, i.e., for each node i,* $|h_{iA_j} - h_{iA_k}| \leq h_{A_jA_k} \leq |h_{iA_j} + h_{iA_k}|;\ \forall\, i \in \mathcal{V};\ \forall\, A_j, A_k \in \mathcal{R}$.

Proof. We first prove the necessary condition, i.e., the resolution set ($\mathbf{P}_{\mathcal{R}}$) of a graph satisfies the above four conditions.

Condition 1: A resolution set uniquely defines each node, and therefore no two distance vectors can be the same.

Condition 2: Consider i^{th} row of $\mathbf{P}_{\mathcal{R}}$, represented by $\mathbf{P}_{\mathcal{R}}(i) = \langle h_{iA_1}, h_{iA_2}, .. , h_{iA_m} \rangle;\ i \in \mathcal{V}$ and $A_k \in \mathcal{M}$. Enrty h_{iA_k} corresponds to the hop-distance of node i to landmark A_k. As, $h_{iA_k} = 0 \iff i = A_k$, i^{th} row $\mathbf{P}_{\mathcal{R}}(i)$, has exactly one entry as '0' if $i = A_k$, i.e., node i is a landmark, and has no entry as '0' if the node is not a landmark.

Similarly, column k in $\mathbf{P}_{\mathcal{R}}$ consists of distance measurements from A_k to each node in graph G. As, $h_{iA_k} = 0 \iff i = A_k$, column k has only one element as '0', which is in the row corresponding to A_k.

Condition 3: Each node satisfies reachability condition as the distance is based on the existing shortest path.

Condition 4: As per triangle inequality [106], for a given triad of nodes A_j, A_k, and i, the distance constraint, $\{h_{iA_j} \leq h_{iA_k} + h_{A_jA_k};\ h_{iA_k} \leq h_{iA_j} + h_{A_jA_k};\ h_{A_jA_k} \leq h_{iA_j} + h_{iA_k}\}$ are satisfied. Thus, $|h_{iA_j} - h_{iA_k}| \leq h_{A_jA_k} \leq |h_{iA_j} + h_{iA_k}|;\ \forall i \in \mathcal{V};\ \forall A_j, A_k \in \mathcal{R}$ is true for a graph and hence is satisfied by a set of valid distance vectors.

Next, to prove the sufficiency of the listed conditions for $\mathbf{P}_{\mathcal{M}}$ to be a $\mathbf{P}_{\mathcal{R}}$ for a graph, we show that if conditions are satisfied by $\mathbf{P}_{\mathcal{M}}$, then there exists a connected graph G of N nodes corresponding to $\mathbf{P}_{\mathcal{M}}$ with $\mathcal{M}$ being its resolution set.

Condition 1: As each DV is unique, it follows that N nodes may be uniquely labeled using different rows.

Condition 2: As each column has exactly one entry of '0', the node labeled with the corresponding row can be designated a landmark. Note that such a row has exactly one '0' entry. As

rows of $\mathbf{P}_{\mathcal{M}}$ may be shuffled without affecting the set of vectors, without loss of generality, we consider landmark k to be represented by k^{th} row. If the i^{th} row has no element as '0', it corresponds to a non-landmark node $i \notin \mathcal{M}$.

Condition 3: The reachability condition, i.e., each node $i \in \mathcal{V}$ is reachable from every landmark $A_k \in \mathcal{M}$, ensures that edges can be added creating paths from every node to every landmark. It follows that each landmark is also connected to every other landmark, resulting in a connected graph.

Condition 4: As $\{h_{iA_j} \leq h_{iA_k} + h_{A_jA_k};\ h_{iA_k} \leq h_{iA_j} + h_{A_jA_k};\ h_{A_jA_k} \leq h_{iA_j} + h_{iA_k}\}; \forall\, i \in \mathcal{V}$ and $A_j, A_k \in \mathcal{M}$, the set of edges added do not violate the triangular inequality.

As each node is uniquely identified by the given set of landmarks and their corresponding distance vectors, and all nodes are connected to each other to form a single connected component then $\mathcal{M}$ is a resolution set and $\exists$ a graph G with DV set $\mathbf{P}_{\mathcal{M}}$. □

The corollary listed below follows from Theorem 3, and may be used to expedite the identification of an ungraphable DV set.

Corollary 0.2. *Let d_k be the maximum element value of column k for a given $\boldsymbol{P}_{\mathcal{R}}$, then the column contains all the integers from '0' to d_k.*

Proof. As each node with distance value d_k is reachable from anchor k, corresponding to i^{th} row in $\mathbf{P}_{\mathcal{R}}$, as per Theorem 3 Condition 3, then $\exists$ a shortest path from A_k to this node of length d_k and therefore, the column has all the elements between '0' and d_k. □

The distance vector set for a resolution set $\mathcal{R}$, $|\mathcal{R}| = m$ and $|\mathcal{V}| = N$, can be restated as follows:

$$\mathbf{P} = \begin{bmatrix} 0 & \dots & h_{1m} \\ \vdots & \ddots & \vdots \\ h_{m1} & \dots & 0 \\ h_{(m+1)1} & \dots & h_{(m+1)m} \\ \vdots & \ddots & \vdots \\ h_{N1} & \dots & h_{mN} \end{bmatrix}_{N \times m} = \begin{bmatrix} L_{m \times m} \\ K_{(N-m) \times m} \end{bmatrix} \tag{4.18}$$

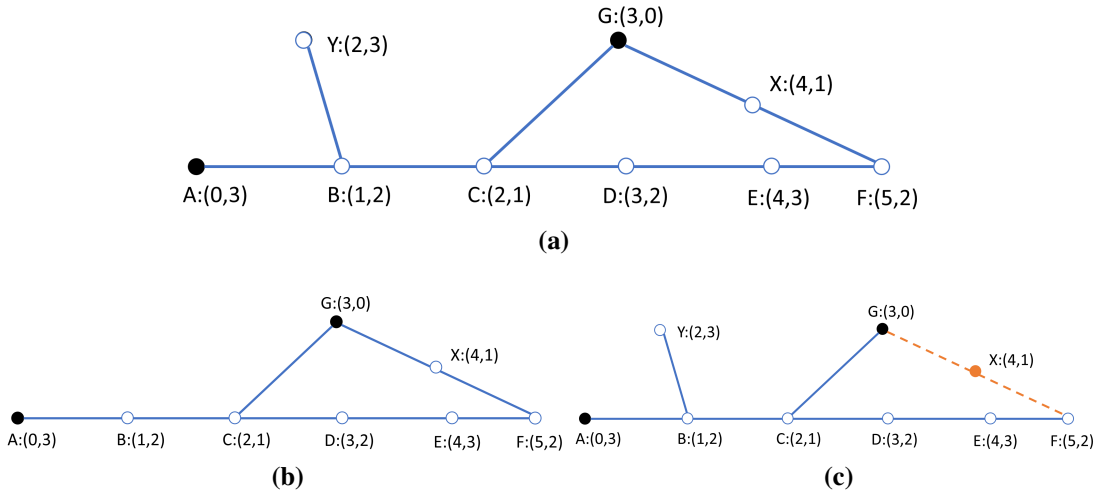


Figure 4.6: Effects of removing distance vectors from $\mathbf{P}_{\mathcal{R}}$: a) Graph for nine nodes and two landmarks (nodes **A** and G) with distance vector set $\mathbf{P}_{\mathcal{R}}$ in Eq. (4.19), b) Eight node graph for $\mathbf{P}_{\mathcal{R}'}$, i.e., with $\langle 2, 3\rangle$ removed from $\mathbf{P}_{\mathcal{R}}$, and c) $\mathbf{P}_{\mathcal{R}''}$ resulting from removing $\langle 4, 1\rangle$ from $\mathbf{P}_{\mathcal{R}}$ is not graphable. $\mathbf{P}_{\mathcal{R}''}$ is thus a deficient DV set and it can be made graphable by adding node X.

where L denotes DVs for landmark nodes and K the DVs for non-landmark nodes. Any graphable $\mathbf{P}_{\mathcal{M}}$ can be expressed in the form of Eq. (4.18).

Next, we look at the case where the distance vectors of some nodes may not be available. For instance, consider the following $\mathbf{P}_{\mathcal{R}}$ corresponding to the graph in Figure 4.6 $[a]$:

$$\mathbf{P}_{\mathcal{R}} = \begin{bmatrix} 0 & 3 \\ 3 & 0 \\ 1 & 2 \\ 2 & 1 \\ 2 & 3 \\ 3 & 2 \\ 4 & 1 \\ 4 & 3 \\ 5 & 2 \end{bmatrix} ; \quad \mathbf{P}_{\mathcal{R}'} = \begin{bmatrix} 0 & 3 \\ 3 & 0 \\ 1 & 2 \\ 2 & 1 \\ \not{2} & \not{3} \\ 3 & 2 \\ 4 & 1 \\ 4 & 3 \\ 5 & 2 \end{bmatrix} ; \quad \mathbf{P}_{\mathcal{R}''} = \begin{bmatrix} 0 & 3 \\ 3 & 0 \\ 1 & 2 \\ 2 & 1 \\ 2 & 3 \\ 3 & 2 \\ \not{4} & \not{1} \\ 4 & 3 \\ 5 & 2 \end{bmatrix} \tag{4.19}$$

$\mathbf{P}_{\mathcal{R}'}$ and $\mathbf{P}_{\mathcal{R}''}$ in Eq. (4.19) have distance vectors $\langle 2,3\rangle$ and $\langle 4,1\rangle$ removed respectively from $\mathbf{P}_{\mathcal{R}}$. $\mathbf{P}_{\mathcal{R}'}$ satisfies all of the four conditions in Theorem 3 for a graphable DV set as shown in Figure 4.6 $[b]$. $\mathbf{P}_{\mathcal{R}''}$ however, does not satisfy one of the conditions, and hence is not graphable.

However, it is possible to make $\mathbf{P}_{\mathcal{R}''}$ graphable by adding the vector $\langle 4,1\rangle$. Thus, the resulting distance vector set is a *deficient* distance vector set where we define a "deficient distance vector set" as follows:

Definition 7. *Deficient distance vector set: The given set of distance vectors $\boldsymbol{P}_{\mathcal{D}}$ is said to be deficient if it is not graphable but $\exists$ a set of DVs $\boldsymbol{P}_{\mathcal{D}'}$ such that, the $\boldsymbol{P}_{\mathcal{D}} \cup \boldsymbol{P}_{\mathcal{D}'}$ is graphable.*

Identification of deficient DV sets is useful in many scenarios dealing with large and complex networks, e.g., when only a subset of DVs are obtainable due to incomplete access, or measurement complexity [79]. While recently developed techniques allow for the recovery of missing elements in DVs of a graphable set [87], no formal methods are currently available for the identification of entire missing DVs. Theorem 3 above provides the necessary and sufficient conditions that have to be met in recovering such missing DVs in a deficient set.

4.7 Reconstruction ambiguity of a resolution set

As shown in Figure 4.2, a set of DVs $\mathbf{P}_{\mathcal{R}} = \{\langle 0,1\rangle, \langle 1,0\rangle, \langle 2,1\rangle, \langle 2,2\rangle, \langle 1,2\rangle\}$ can be reconstructed as graphs in Figure 4.2 $[a]$ or $[b]$. $\mathbf{P}_{\mathcal{R}}$ does not guarantee an exact reconstruction of G from $\mathcal{R}$ as $\mathcal{C}$ does. Thus, reconstructing G from $\mathbf{P}_{\mathcal{R}}$ may lead to multiple solutions. The higher the number of possible graph solutions, the higher the reconstruction ambiguity of $\mathbf{P}_{\mathcal{R}}$. In this section, we introduce the reconstruction ambiguity of resolution sets as well as the notions of maximum and minimum edge graphs.

Reconstruction of a graph from $\mathbf{P}_{\mathcal{R}}$ involves the derivation of entries of the adjacency matrix, $\mathbf{A}$. While many of its entries can be derived as '0' or '1' using the properties of the distance vectors, the ambiguity is introduced by the entries which can be set to either '0' or '1' without violating the

DVs. As $\delta_{ij} = 1$ is a necessary condition for adjacency of nodes i and j, when this condition is not satisfied the nodes cannot be adjacent. This yields the following sufficient condition:

$$\mathbf{A}_{ij} = \mathbf{A}_{ji} = \begin{cases} 0; \ \textit{if } \delta_{ij} > 1 \end{cases} \tag{4.20}$$

For node i, the set of possible neighbor nodes, $\ominus_i$, can be defined as

$$\ominus_i = \{j \mid \delta_{ij} = 1\} \tag{4.21}$$

Definition 8. *Maximum edge graph: A graph with edges between node i and all the nodes in $\ominus_i \ \forall \ i$ is said to be a 'maximum edge graph' for the given $\boldsymbol{P}_{\mathcal{R}}$. Let the set of edges in this graph be denoted as $\mathcal{E}_{max}$.*

A valid graph may in many cases be constructed with edges between node i and only some of the nodes in $\ominus_i$. There may be a subset of nodes in $\ominus_i$ that have to be connected to node i to satisfy the reachability of node i from the given landmark A_k as given in Definition 6.

Definition 9. *Minimum edge graph: A graph is said to be a 'minimum edge graph' for the given $\boldsymbol{P}_{\mathcal{R}}$ if there is no other connected graph with a smaller number of edges that satisfies $\boldsymbol{P}_{\mathcal{R}}$. Let the set of edges in this graph be denoted as $\mathcal{E}_{min}$.*

Following two rules can be used to identify a minimum edge graph corresponding to $\mathbf{P}_{\mathcal{R}}$:

1. As per the reachability definition for node i, $\exists \ j$ such that, $j \in \ominus_i$ and $(h_{jA_k} = h_{iA_k} - 1)$, i.e., node j lies on the shortest path from landmark A_k to node i. Denote the set of such nodes j by ψ_{iA_k} and if $|\psi_{iA_k}| = 1$ then node j, $j \in \psi_{iA_k}$, is definitely a neighbor of node i, i.e., $\exists$ an edge (i, j) in graph G.

$$\mathbf{A}_{ij} = \mathbf{A}_{ji} = \begin{cases} 1; \ \textit{if } j \in \psi_{iA_k} \textit{ and } |\psi_{iA_k}| = 1 \end{cases} \tag{4.22}$$

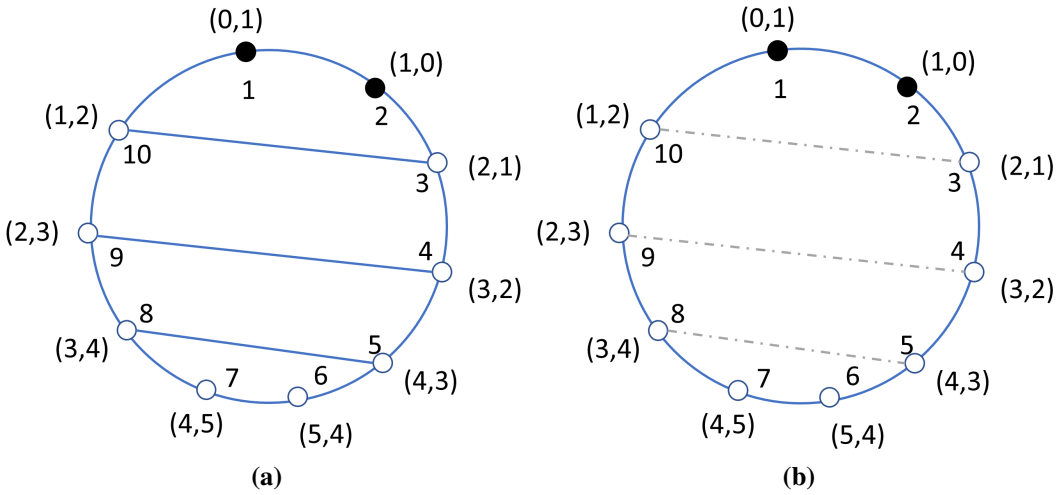


Figure 4.7: Maximum and minimum edge graphs for an example distance vector set: a) Maximum edge graph, b) Minimum edge graph shown in solid lines. The three ambiguous edges are indicated by dashed lines, which indicate an ambiguity range of 8.

2. Add the minimum number of remaining unassigned elements in $\psi_{iA_k} \forall\, i, A_k$ that are needed to ensure reachability of every node from each anchor, i.e., $\mathbf{A}_{ij} = \mathbf{A}_{ji} = 1$ for at least one j in ψ_{iA_k} corresponding to each node i.

Though rules 1 and 2 stated above help us determine the minimum set of entries of $\mathbf{A}$ to get a connected graph for the given $\mathbf{P}_{\mathcal{R}}$, all the remaining entries, other than those identified as '0' by Eq. (4.20), in $\mathbf{A}$ are ambiguous, i.e., undetermined whether $\mathbf{A}_{ij} = 0$ or 1 based on the given $\mathbf{P}_{\mathcal{R}}$.

The difference between the sets of edges of maximum and minimum edge graphs for $\mathbf{P}_{\mathcal{R}}$ is given by the set difference $\mathcal{E}_{max} \setminus \mathcal{E}_{min}$. Then the number of ambiguous edges , τ, can be given by following equation:

$$\tau = |\mathcal{E}_{max} \setminus \mathcal{E}_{min}| \tag{4.23}$$

There are 2^{τ} possible graphs for the given $\mathbf{P}_{\mathcal{R}}$ which is the ambiguity range of reconstructing the graph from the given distance vector set.

Figure 4.7 illustrates minimal and maximum connectivity of a graph for the same given set of distance vectors. Note that all the combinations of ambiguous edges lead to a connected graph for the given DV set. Thus, all the different combinations of possible networks that fall between the maximum and minimum edge network are nothing but the *'Range of Ambiguity'* for recovering a

Table 4.2: Metric dimension and Link dimension distributions for all the graphs with size 4,5, and 6 respectively.

$\lvert\beta(G)\rvert$ or $\lvert\xi(G)\rvert$	$\Sigma_\beta(4)$	$\Sigma_\xi(4)$	$\Sigma_\beta(5)$	$\Sigma_\xi(5)$	$\Sigma_\beta(6)$	$\Sigma_\xi(6)$
1	1	1	1	1	1	1
2	4	4	14	8	62	24
3	1	1	5	11	42	66
4	-	-	1	1	6	20
5	-	-	-	-	1	1

graph from a given distance vector set. The ambiguity can vary depending on the type of network. Some DV sets may have no ambiguity while others may have several edges taking the predicted network away from a unique solution. A resolution set can thus be used to represent a set of graphs that vary only in the edges which fall in the ambiguity range of $\mathbf{P}_{\mathcal{R}}$.

4.8 Graph distribution over metric dimension and link dimension

In this section, we summarize the results for metric dimension and link dimension for all possible graphs with $N = 4, 5,$ and 6 respectively. Let $\Sigma_\beta(N)$ denote the number of graphs with metric dimension β and $\Sigma_\xi(N)$ denote the number of graphs with link dimension ξ for graph of size N. Table 4.2 lists the number of graphs for a given value of N that have a particular value of metric dimension or link dimension. For example, for $N = 5$ there are 14 graphs with metric dimension, $\beta(G) = 2$ and eight graphs with link dimension, $\xi(G) = 2$. For $N = 4$, the link dimension and metric dimension distributions are identical for graphs with four nodes. As N increases, it is more and more likely that the link dimension is higher than the metric dimension.

Correlation between spectra moments and graph dimensions:

Spectral graph theory, also referred to as 'graph spectra', is the study of the properties of a graph in relationship to the characteristic polynomial, eigenvalues, and eigenvectors of matrices associated with the graph, such as its adjacency matrix or Laplacian matrix. It provides useful

information for various applications such as internet topology, pattern recognition, and computer vision [107]. Thus, we also examined metric dimension and link dimension in relation to graph spectra. We focus on spectral moments and energy of a graph, which are frequently studied quantities in the context of graph spectra.

The spectral moments $S_k(k = 1, 2, \dots)$ of a graph G, which are defined as the sum over the k^{th} powers of all eigenvalues of the adjacency matrix **A**, have been used to classify graphs and have a lot of interesting properties [108]. The Pearson's correlation coefficients between link dimensions and spectral moments for all graphs of size six, hold values ≤ 0.5, depicting a weak correlation between link dimension and spectral moments. The energy, $E(G)$, of a simple graph G is defined to be the sum of the absolute values of the eigenvalues of G [109] which tend to predict the strongly correlated betweenness and the eigencentrality of vertices in large complex networks, which are expensive to calculate and requires global processing of a network [110].

Table 4.3 lists Pearson's correlation coefficients between graph dimensions and spectral moments. With values ≤ 0.5, it can be observed that link dimension shows a very weak correlation between all of the spectral moments.

Table 4.3: Pearson's correlation coefficients between Metric dimension and Link dimension and the spectral moments of graphs for $N = 6$

Spectral moment	$\xi(G)$	$\beta(G)$
S_2	0.4337	0.5139
S_3	0.3745	0.5371
S_4	0.4529	0.6283
S_5	0.3826	0.5604
S_6	0.4122	0.600
number of edges	0.433	0.513
number of triangles	0.374	0.537
(number of edges - number of triangles)	-0.2072	-0.428

The energy, $E(G)$, of a simple graph G is defined to be the sum of the absolute values of the eigen values of G [109]. We observed that the energy of the graph with $N = 6$ has a Pearson's correlation coefficient of 0.14 with link dimension and 0.025 with metric dimension. This further

bolsters our previous observation that graph eigen values and graph dimensions have a very weak correlation. Thus, highlighting the lack of overlap of information captured by the different concepts namely graph dimensions and graph spectra.

While it is true that $\xi(G) \geq \beta(G)$, the Pearson's correlation coefficient between β and ξ for graphs with $N = 6$ turned out to be 0.67 indicating a lack of strong correlation between the two.

4.9 Summary

Dimensionality in graphs, such as metric dimension and edge metric dimension have focused on the dimensionality of a set of distance vectors to uniquely label nodes or edges in a graph. They are only aimed at distinguishing nodes or edges in a graph, and as a result, it is not possible in general to identify or reconstruct the graph from such a vector set. We introduced the novel concepts of link dimension and construction set, which while allowing for unique labeling of edges and nodes, also allow for its unique identification and reconstruction. Essentially, the Distance Vector (DV) set corresponding to the construction set is a formal and exact representation of G. The DV set is a very compact ($N \times \xi(G)$ matrix with $\xi(G) << N$ as opposed to $N \times N$ for adjacency and distance matrices) for many graphs or graph families. This paper, therefore, extends the distance vector based approach beyond unique labeling of nodes to unique identification of graphs.

Furthermore, we address the inverse relationship, i.e., that of a given set of positive integer vectors to graphs. The conditions for graphability of a distance vector set are presented, together with the associate concepts of ambiguous edges, invisible edges and ambiguity range for a graphable DV set, as well as deficient vector sets that may be converted to graphs by adding new vectors.

Link dimension, as well as bounds for link dimension for several graph families, are provided which satisfies the relation of link dimension being greater than or equal to metric dimension for the given graph. A basic algorithm for deriving a construction set from a resolution set is presented along with the necessary and sufficient conditions for a resolution set to be a construction set.

This research introduces a new question of "How to uniquely reconstruct a graph from its selected distance vectors?" and offers a solution in the form of "Link Dimension". We further validate the work done in this area that showcases gaps in reconstructing a graph using its resolution set. The ability to go from vectors to graphs with unique graphability, compared to the traditional approach of labeling nodes with vectors, opens several novel problems for investigation.

Link dimension for different families of graphs that are not provided in this work can be derived in the future. Extending the concept of link dimension to larger real-world networks and developing scalable algorithms based on compact construction sets instead of the adjacency or distance matrices can be further investigated upon.

Chapter 5

Topology Extraction from Partial Network Data using Low-rank Matrix Completion

For many important network types (e.g., sensor networks in complex harsh environments and social networks) physical coordinate systems (e.g., Cartesian), and physical distances (e.g., Euclidean), are either difficult to discern or inapplicable. Accordingly, coordinate systems and characterizations based on hop-distance measurements, such as Topology Preserving Maps (TPMs) and Virtual Coordinate (VC) systems are attractive alternatives to Cartesian coordinates for many network algorithms. Herein, we present an approach to recover geometric and topological properties of a network with a small set of distance measurements. In particular, our approach is a combination of shortest path (often called geodesic) recovery concepts and low-rank matrix completion, generalized to the case of hop-distances in graphs. Results for sensor networks embedded in 2D and 3D spaces, as well as social networks, indicate that the method can accurately capture the network connectivity with a small set of measurements. TPM generation can now also be based on various context appropriate measurements or VC systems, as long as they characterize different nodes by distances to small sets of random nodes (instead of a set of global anchors). The proposed method is a significant generalization that allows the topology to be extracted from a random set of graph shortest paths, making it applicable in contexts such as social networks where VC generation may not be possible.

5.1 Introduction

Spatial networks are often localized using physical coordinate systems (e.g., Cartesian), and physical distances (e.g., Euclidean). Although applicable, these localization schemes are expensive – due to the GPS costs, computationally heavy – due to latitude and longitude numbering system, and can be dependent on the technology for accuracy. Accordingly, coordinate systems

and characterizations based on hop-distance measurements, such as Topology Preserving Maps (TPMs) and Virtual-Coordinate (VC) systems are attractive alternatives to Cartesian coordinates for many network algorithms as a foundation to work on.

Large and complex networks naturally arise in communication networks, the Internet-of-things (IoT), and many other systems of importance. Data embedded in such networks exhibit distributed, intricate, and dynamic patterns. Our ability to extract information from and about these networks, detect anomalies, and influence their performance can be substantially improved by a deeper understanding of their local and global structures [111–113]. Yet such operations are often impeded due to the size and complexity of these networks, and constraints such as energy, measurement cost, and accessibility that prevent *geometric and topological* structure of the entire network from being fully observed, measured, characterized, or processed. Inferences have to be made and patterns should be detected in the absence of complete information. Herein, we present a novel approach to recover geometric and topological properties of a network, e.g., distances, topology, connectivity, and layout, with a small set of distance measurements. Our approach is a combination of shortest path (often called geodesic) recovery concepts and low-rank matrix completion, generalized to the case of hop-distances in graphs.

The proposed method is scalable in its computation and communication, as well as graceful in its degradation in the presence of limited measurements. In particular, we sample a network with a small set of pair-wise hop-distances and use matrix completion techniques to capture the topology of the network. We demonstrate the technique by deriving topology preserving maps indicative of the layout of 2D and 3D sensor networks with far fewer measurements compared to existing schemes. Results indicate that the method can accurately capture the network connectivity with a small set of measurements. TPM generation can now also be based on various context-appropriate measurements or VC systems, as long as they characterize different nodes by distances to small sets of random nodes (instead of a set of global anchors).

Furthermore, this new approach extends the applicability of hop-distance-based techniques to cover complex networks such as social networks where other distance measures, such as Euclidean

distances, are either not available or do not make sense while providing a foundation for using a broader set of sampling strategies.

In a nutshell, our approach can be described as follows:

1. **Choose a representation for a network** (or graph). We choose to focus on hop-distance matrices, **H**, instead of the adjacency matrices **A**. This is because unlike adjacency matrices, hop-distance matrices are found to be of low-rank, especially for real-world networks

2. **Use a sampling scheme to obtain a small set of elements of hop-distance matrix H.** We sample a small percentage of entries from the complete hop-distance matrix for experimentation. Sampling is performed by selecting M anchor nodes from total N nodes in the network, such that $M << N$. Then, we measure hop-distances from each anchor node to a fraction of other nodes in the network. The sampled entries are called as 'Observed' and the deleted entries are called as 'Unobserved'.

3. **Fill in the incomplete, sampled hop-distance matrix H.** The sampled entries are processes and used to extract topology information of the network. Leveraging the low-rankness of the hop-distance matrix, low-rank matrix completion technique is used to predict the missing hop-distances. Unobserved entries are predicted and prediction performance is evaluated against the actual hop-distance entries.

As far as we are aware, using anchor nodes to generate partially observed measurements of hop-distance matrices and then completing them using a low-rank assumption is a novel problem formulation.

Section 5.2 states the motivation behind this work. Literature related to this work is reviewed in Section 5.3. Section 5.4 talks about the important contributions of this work. The theoretical considerations and results supporting proposed geodesic sampling schemes and low-rankness are outlined in Section 5.5. Methodology for VC based sampling and reconstruction is presented in Section 5.6 followed by results in Section 7.4 and conclusion in Section 5.8.

5.2 Motivation

Of specific interest to us are networks of inexpensive wireless devices such as smart Radio-Frequency Identification (RFID) tags or self-powered sensor nodes embedded in complex 2D or 3D surfaces and volumes. Here, the network features are often characterized using the physical (Cartesian) coordinates of each node, or the corresponding Euclidean Distance Matrix (EDM) that specifies the Euclidean distances among node pairs. One then relies on the Euclidean properties of the network layout (e.g., the distance between nodes) for functions such as area or volume coverage, topology control, sensing, and routing. Physical location estimation is based on measurements such as the Received Signal Strength Indicator (RSSI) and time delay [114]. In networks deployed in harsh or complex environments, such methods are hampered or made completely ineffective by issues such as multi-path interference, reflections, shadowing and clock synchronization [82, 115]. Although physical coordinates provide a meaningful representation for node location, for sensor networks deployed on complex surfaces and volumes, *graph distance* based techniques that use only connectivity measurements offer a more robust, and attractive alternative [82]. For instance, in routing they overcome local minima caused by concave physical voids.

Connectivity based techniques are also generalizable to networks such as social networks, for which there is no notion of a location in a physical or Euclidean space. Even if a node is associated with a location, physical proximity does not necessarily imply connectivity. Two nodes separated by a reflective metal surface or an absorbing material need to communicate via intermediate nodes. In fact for many such applications, what matters is the logical topology, i.e., connectivity and the hop-distances among node pairs.

While sensor networks differ in aspects such as physical distribution and the dependence of communication topology on physical distance, both types of networks can be represented in the form of nodes and edges in a graph. Furthermore, the node connectivity, characterized by network topology, plays a key role in both these networks. The solution we propose is applicable in both of these domains, as well as many other problems that can be represented as graphs, such as protein interaction networks and disease spread patterns.

5.3 Related Work

This section reviews hop-distance based network sampling schemes, the virtual-coordinate based representation, and their relationship to topology coordinates which recover the Euclidean properties of 2D and 3D networks. An anchor-based VCS is an M-dimensional abstraction of the network connectivity, where each node is represented by an M tuple, called VC vector which contains the shortest path length (in hops) from the node to each of a set of M anchors [116, 117]. In an Aligned VC system [118], each node re-evaluates its coordinates by averaging its own coordinates with its neighbors' coordinates. Axis-based Virtual Coordinate Assignment Protocol [119] estimates a 5-tuple VC for each node corresponding to longitude, latitude, ripple, up, and down.

A key question related to anchor-based VCS is the number M and the placement of anchors. If an adequate number of anchors are not appropriately deployed, it may cause the network to suffer from identical coordinates and local minima [120], resulting in logical/virtual voids. The overhead associated with time and energy consumed for coordinate generation grows with the number of anchors. The difficulty of determining the optimal anchor set is compounded by the fact that the number of anchors and their optimal placement are dependent on each other. In Virtual Coordinate Assignment Protocol, the coordinates are defined based on three anchors [117], while all the perimeter nodes are selected in [118]. Extreme Node Search uses two initial random anchors which allow the selection of extreme nodes on internal and external boundaries as anchors [86]. An attractive alternative is to bypass anchor selection altogether and select a set of random anchors. Note, while from a networking perspective a random selection of anchors may seem somewhat odd (i.e., a judicious placement might be viewed as more appropriate), from a matrix completion perspective such a placement is not only justified, but in many instances may be optimal [51, 121–123]. Again, such network organizational ideas have a dual perspective in the mathematical literature and the selection of anchors is closely related to the idea of *incoherence* in low-rank matrix recovery literature [51, 121–123].

Geographical features such as boundaries and voids are missing from an anchor-based VCS. In our previous work [82, 85], recovery of geographic features from VCS is achieved by TPMs.

TPMs derived from VCs are maps that are nearly homeomorphic to physical maps [82]. A TPM is a distorted version of the real physical layout (map) in such a way that the distortion accounts for connectivity information. In the case of a 2D or 3D sensor network with M anchors, VCS is a mapping from the 2D or 3D network layout to an M-dimensional space. TPMs recover a 2D or 3D projection from this M-dimensional representation such that it preserves the main features such as boundaries and shapes of the network. Thus, TPMs can serve as an effective alternative for physical coordinates in many network related functions. Reconstruction of the graph from its VCs is attempted in [88] using an algorithmic approach. TC based schemes have demonstrated performance comparable, or better than the corresponding geographic coordinate based counterparts [85, 124, 125].

While the algorithms that are based on VCs or TCs [82, 126] provide a viable, competitive, and robust alternative to traditional geographic coordinate based methods, these techniques have so far required the complete set of VCs to extract the geometric information. However, as we demonstrate here, one can recover topological properties of a network *without the need for complete knowledge of the virtual coordinates*. While the complexity of generating a complete distance matrix through flooding is of order $O(N^2)$, that for VC generation from a set of $M (M << N)$ anchors is only $O(NM)$. Herein, we further reduce this computational cost and justify those results. In particular, we demonstrate a connection between TPMs, NDR (Nonlinear Dimension Reduction) [127, 128], and low-rank matrix completion problems [51, 121–123, 129].

Finally, we observe that the computation of a low-rank approximation of an EDM using PCA is equivalent to the Multi-Dimensional Scaling (MDS) [130, 131] algorithm. In other words, given noisy measurements of all Euclidean distances between a set of points, MDS provides a way of computing the relative positions of those points. Even closer to our proposed technique, many authors consider *geodesic* adjacency matrices $\mathbf{A}$ [127, 128] generated by drawing short range, or *neighbor*, distances from the EDM ($\mathbf{D}$). They then compute *long range* distances by way of *shortest path distances in the graph represented by* $\boldsymbol{A}$. In particular, one might be in the position where small Euclidean distances between points can be accurately measured, but distances between far-

away points are not known. One can represent such a set of points by a graph where points are linked by edges if they are close together, but edges are absent for points that are far away from each other. A low-rank approximation of such a graph shortest path based distance matrix is equivalent to the Isomap [127, 128] algorithm for NDR. An MDS based algorithm is proposed in [132] for estimating connectivity from incomplete passive measurements, which we later compare against the proposed method.

5.4 Contribution and impact

The main contribution of this paper is a technique that combines VC based techniques with low-rank matrix completion, which allows the extraction of topology and geometric features of a network from a small set of shortest path distances. Specifically, there are two outcomes.

- First, in the case of networks embedded in 2D and 3D physical spaces, we demonstrate that the topology preserving maps can be recovered using only a small fraction of VC values, as opposed to existing method [82] that requires the full set of VCs.

- Second, we broaden the possible set of network sampling schemes far beyond our previous results in [78, 82, 91], and point to a theory on which new sampling schemes may be grounded, i.e., selection of samples should not hinder the ability to complete the resulting incomplete low-rank matrices. We demonstrate the reconstruction of the network topology from a small set of shortest path length measurements. Consequently, TPMs can now be generated as well as topology/connectivity of networks can be extracted based on a variety of geodesic measurement approaches, of which an anchor-based VCS is only one instance.

The novelty and the significance of this work may be gauged by the following impacts it can have on network analytics:

- The theory of low-rank matrix completion can now be applied for exploiting the sparseness of complex real-world networks, and to develop communication and computation efficient techniques for large-scale networks. Although $\mathbf{H}$ for an arbitrary graph may not be low-rank,

we demonstrate that for a broad class of complex real-life networks such distance matrices are surprisingly low-rank. Our analysis is based upon ideas in low-rank matrix completion [51, 121–123, 129], which have been shown to scale to large problems with many thousands, if not millions, of entries [123, 129], and the underlying topology that we uncover is closely related to ideas in Non-linear Dimension Reduction (NDR), such as Isomap [127, 128], but generalized to the case where only hop-distances are measured.

- The applicability of hop-distance based property extraction is extended to cases where certain distances and connectivities are not observable. For large and complex networks, and especially those with access limitations, we can likely not even measure a complete set of hop-distances to a set of common anchors. Examples of such cases include sensor and communication networks with restricted access to certain nodes. As explained in [79] using examples from communication networks, it is realistic to obtain the distances between nodes in many cases, while it is difficult or impossible to obtain information about edges or absence of edges that are far away from the query node. The same argument extends to many practical problems dealing with data, e.g., pathway prediction problem in proteins [133, 134], where the distance between two nodes (proteins) can be evaluated yet finding the shortest path by some measure (e.g., folding sequence) from one node to the other is complex.

- Matrix completion algorithm finds many applications such as, link prediction and finding top N recommends for users [44] [135]. While matrix completion as applied to Euclidean distance matrices is not new [136–140] our approach differs in a key aspect from those currently found in the literature. Our focus is on using *hop-distances* rather than the more classic distance measures such as Euclidean distances [141]. In other words, instead of considering Euclidean Distance Matrices (EDM) where each entry of the matrix codes for the Euclidean distance between two nodes, we consider HDM where each entry of the matrix codes for the *graph shortest path distance* between two nodes. Note, as far as the authors are aware, the idea of using low-rank matrix completion on graphs and networks where only hop-distances are available is novel to this research. Accordingly, the literature and methods

for EDM completion [136–140] are not applicable in this context since we treat problems where Euclidean distances are assumed to not be available or applicable, such as in social networks.

Accordingly, we leverage HDMs, which are the most faithful representations of the network to which we have access. As far as we are aware, problems of such generality have not been considered before.

It is important to note that HDMs and EDMs can be quite different, even for the same set of points. In particular, there may be many EDMs that are associated with the same HDM. For example, in Figure 5.1 we show two networks that give rise to identical HDMs, with one of the EDMs admitting a low-dimensional representation while the other EDM does not.

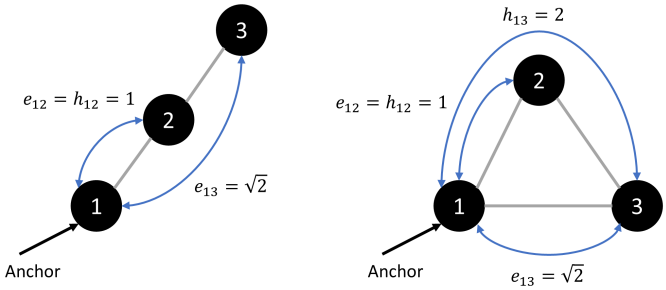


Figure 5.1: Two network layouts with the same HDM but different EDMs. The HDM is computed from the EDM using the simple rule that two nodes communicate if their Euclidean distance is less than or equal to 1. The HDM is low-rank since the points can be embedded on a one-dimensional line and still exactly preserve their hop-distances. However, the right EDM is not low-rank, since there is no projection of these points onto a line that can preserve all of the Euclidean distances between the points.

5.5 Theory

In order to solve the problem of accurately extracting topological information about a network, we focus on it's two important features, namely a) measurement of the network in terms of hop-distances and b) assuming that the networks we work with portray low-rankness in their hop-distance matrices. These two features are very important assumptions of this work.

- **Measuring a network as its hop-distance matrix:** We focus on hop-distances while measuring networks because unlike adjacency matrices, hop-distances capture global attributes about a network along with local connectivity. They also follow constraints such as Triangle-Inequality [142] which allows one to derive bounds on other hop-distances in the network.

- **Low-rankness of hop-distance matrices:** Several real-world networks are low-rank in their hop-distance matrices. We leverage this quality of networks and hop-distance matrices to extract the characteristics of the original network from partial measurements. Applying Low-rank Matrix Completion is also more effective on low-rank data.

While many papers analyze low-rank EDMs [138, 140], it is not clear *a priori* that a low-rank assumption for HDMs is at all appropriate. However, in this Section, we provide two flavors of results to justify considering HDMs from a low-rank point of view. First, we note that there is a body of theory, perhaps less well known than the similar body of theory for EDMS, that provides guarantees on the low-rank properties of HDMs for certain classes of graphs. Second, we look at some example real-world networks and note that their HDMs are indeed low-rank when the entire matrix is observed. Accordingly, it is not unreasonable, from both empirical and theoretical perspectives, to approach HDM completion from a low-rank matrix completion perspective.

5.5.1 Data representation and hop-distance based sampling

We focus our attention on *undirected graphs* G defined by $G = \{V, E\}$, where V is the set of nodes or vertices and E is a set of edges corresponding to communication links, etc. Such a graph may be represented by an *adjacency matrix* $\mathbf{A}$ [143], where

$$\mathbf{A} = [a_{ij} | a_{ij} = 1 \text{ if } (i,j) \in E,\ 0 \ \text{ otherwise }]\,. \tag{5.1}$$

Note that $a_{ij} = a_{ji}$ when G is undirected. To make the current text concise, herein we focus on *unweighted* graphs where each $a_{ij} \in \{0, 1\}$. Our main object of study will be *Hop-Distance* Matrices (HDMs), $\mathbf{H} \in \mathbb{N}_0^{N \times N}$, (where $\mathbb{N}_0$ denotes the non-negative integers $\mathbb{N} \cup \{0\}$), where

$$h_{ij} = \text{the length of the shortest path from node } i \text{ to node } j. \tag{5.2}$$

H, by construction, is symmetric and is invariant to the situation where two nodes have multiple paths between them of the same shortest length. From a mathematical perspective, hop-distance h_{ij} may be thought of as the computation of *geodesic* (or shortest path) in a graph [127, 128, 143]. In particular, herein the term *geodesic* will refer to a shortest path between nodes, either when embedded in a particular space (e.g., the sum of straight line distances between node pairs forming a path in a Euclidean space), or when considered as a node in a graph (e.g., as computed using Dijkstra's algorithm [144]).

Our goal is to capture the topology accurately using efficient network sampling schemes. Prominent among hop-distance based sampling methods are those relying on anchor-based Virtual Coordinate Systems (VCS), in which each node is represented by a Virtual Coordinate (VC) vector corresponding to the *minimum* hop-distances to a small set of nodes, known as *anchor nodes* [117–119]. A simple graph with one anchor is shown in Figure 5.2. VCs are generated, e.g., in wireless sensor networks, by each anchor initiating a packet that gets flooded in the network. The packet contains a counter that gets incremented at each node thus indicating the hop-distance it has traveled. A node uses the lowest value of the distance from a particular anchor as the corresponding coordinate. Hop-distance thus can be measured accurately as it depends only on the communication topology, in contrast to Euclidean distance measurements that rely on analog measurements such as signal strength or time delay, which are highly errorprone due to factors such as fading, multipath, interference and synchronization errors [83, 115]. If we have M anchor nodes, then a VCS is an $N \times M$ sub-matrix **P** of **H** where the i-th row provides an M dimensional coordinate vector for the i-th node in the graph.

A VCS is a "relative" coordinate system, as opposed to classic "absolute" coordinate systems such as Cartesian or Spherical. In particular, a VCS does not possess, or require, an absolute origin or other fixed geometric properties such as a notion of angle. However, a VCS lacks directional information as each coordinate indicates only the distance to an anchor. Thus all the information about the network layout such as shapes, voids, and boundaries are lost in a VCS. This issue leads to derivative coordinates such as Topology Coordinates (TCs), where one performs an eigen-decomposition of **P** or **H** similar to that performed in Principal Component Analysis (PCA) [124, 125], and Topology Preserving Maps (TPMs) that use most significant principal components to generate 2D or 3D maps [82]. TPMs recover the general shapes and boundaries of the physical network layout, thus providing an effective alternative for representing 2D and 3D sensor networks and carrying out operations such as routing and boundary detection, but without the need for physical distance measurements [124, 125].

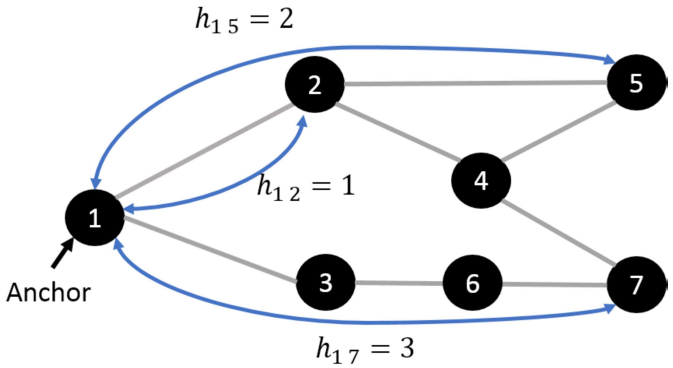


Figure 5.2: A simple example of a graph with an anchor node. In this graph node-1 is an anchor node and we have measured three distances, namely $h_{12} = 1$, $h_{15} = 2$, and $h_{17} = 3$. Note, that knowing a long distance does not imply that the shorter distances along the shortest path are also known. For example, while h_{17} is known we do not assume that h_{13} or h_{16} are necessarily known.

5.5.2 Anchor based VCs

We follow the notation in [85] and consider networks where M of the N nodes are designated as "anchors". With an anchor-based VCS, each of the N nodes in the network is characterized by

a VC vector of length M, i.e., each node is labeled by its shortest-path hop-distance to each of the M anchors.

Let $\mathbf{P} \in \mathbb{N}_0^{N \times M}$ be the matrix containing the VCs of all the nodes, e.g., the i-th row corresponds to the $\mathbb{N}_0^{1 \times M}$ VC vector of the i-th node, and j-th column corresponds to the j-th virtual coordinate of all the nodes in the network, i.e., with respect to j-th anchor.

This matrix can be written as

$$\mathbf{P} = \begin{bmatrix} h_{1A_1} & \dots & h_{1A_M} \\ \vdots & \ddots & \vdots \\ h_{NA_1} & \dots & h_{NA_M} \end{bmatrix} \tag{5.3}$$

where h_{iA_j} is the hop-distance from node i to anchor A_j. $\mathbf{P}$ is precisely a subset of the full hop-distance matrix $\mathbf{H}$ derived by selecting just a few anchor nodes and constructing $\mathbf{P}$ from the columns corresponding to those anchor nodes. For large networks it is generally desirable to have only a small subset of nodes as anchors, i.e., $M \ll N$.

In particular, one can equivalently think of $\mathbf{P}$ as a (non-principal) *sub-matrix* of the full hop-distance matrix $\mathbf{H}$. If we decompose $\mathbf{H}$ into blocks by writing

$$\mathbf{H} = \begin{bmatrix} \mathbf{A} & \mathbf{B}^T \\ \mathbf{B} & \mathbf{C} \end{bmatrix}, \tag{5.4}$$

then $A \in \mathbb{N}_0^{M \times M}$ contains the hop-distances between the M anchors and themselves, $B \in \mathbb{N}_0^{(N-M) \times M}$ contains the hop-distances between the M anchors and the $N - M$ non-anchor nodes, and $C \in \mathbb{N}_0^{(N-M) \times (N-M)}$ contains the hop-distances between the $N - M$ non-anchor nodes and themselves, which in our case are missing entries. In this way, $\mathbf{P}$ can equivalently be written as

$$\mathbf{P} = \begin{bmatrix} \mathbf{A} \\ \mathbf{B} \end{bmatrix}. \tag{5.5}$$

Note, the prediction of both **P** and **H** from partial observations are of interest in the current context. Accordingly, in Section 7.4, we will provide numerical results for both problems.

5.5.3 Low-rank matrices

As can be inferred from our prior work [82], and as we further demonstrated above, hop-distance matrices **H** for many interesting and realistic networks are, somewhat surprisingly, *approximately low-rank*. It is this empirical observation that inspires our work.

A widely used tool for analyzing low-rank structure in matrices is the Singular Value Decomposition (SVD) [145] and the closely related idea of Principal Component Analysis [146]. In particular, given our VC matrix $\mathbf{P} \in \mathbb{R}^{N \times M}$ one can write **P** as

$$\mathbf{P} = \mathbf{U}\Sigma\mathbf{V}^T \tag{5.6}$$

where, assuming $M < N$ as $U \in \mathbb{R}^{N \times M}$, $\Sigma \in \mathbb{R}^{M \times M}$, and $V \in \mathbb{R}^{M \times M}$. In addition, the columns of U and V are orthonormal (i.e., U and V each are sub-matrices of a *unitary* matrix) and Σ is diagonal. Finally, the diagonal entries of Σ are called the singular values of **P** and the rank of **P** is precisely the number of non-zero singular values.

Accordingly, one can compute an approximation of **P** by setting the "small" entries of Σ to 0, using an appropriate threshold, to generate an approximation $\Sigma \approx \hat{\Sigma}$. **P** can then be approximated similarly by setting $\mathbf{P} \approx \hat{\mathbf{P}} = U\hat{\Sigma}V^T$. Such ideas have a long history, with an important milestone being the seminal work of Eckart and Young in 1936 [145].

5.5.4 Graph reconstruction: Adjacency matrices

Given partial knowledge of a graph, there are several techniques for predicting the unknown information about the graph, including an array of combinatorial techniques. Examples include NP-hard combinatorial algorithms such as minimal Hamiltonian completions [147, 148] (i.e., adding the minimum number of edges to a graph to make a Hamiltonian path that visits each vertex once) and minimal Chordal completions [149] (i.e., adding the minimum number of edges to a graph

such that every cycle on four, or more, nodes has a chord). However, with the advent of effective and scalable tools for *large-scale low-rank matrix* completion, in our work we choose to focus on matrix completion techniques that make a *low-rank assumption*.

Of course, given the natural connection between adjacency matrices and graphs, one would be tempted to look at low-rank structures in matrices such as **A**. Much is known in such cases, including the facts that [150]:

- The only rank-0 adjacency matrix is for the graph with no edges.
- The only rank-1 adjacency matrix is the complete graph.
- The only rank-2 adjacency matrix is the complete bipartite graph (and complete tripartite is rank-3, etc.).

It is also known, for example, that for a subgraph $S \subseteq G\ rank(S) \leq rank(G)$ [150].

Note, *Graph Laplacians* is an area in which the low-rank structure of graphs can be precisely defined and is well understood [151]. We merely observe that given an adjacency matrix **A** one can define the diagonal matrix $\mathbf{D}_{row}$ as the row sums of **A**. Given such a $\mathbf{D}_{row}$, a Graph Laplacian for **A** is defined as $\mathbf{L} = \mathbf{D}_{row} - A$. It is well known that the Graph Laplacian is low-rank if, and only if, the underlying graph is disconnected [151]. Unfortunately, such techniques do not lead to the types of predictions that we desire.

5.5.5 Approximately low-rank HDMs

Low-rank matrix completion has been used in Euclidian Distance Matrices (EDMs) [138, 140] and have been proved to be low-rank in nature [138]. However, as we saw that not all networks can be represented in terms of Euclidian distances, for example communication networks, software modules, and web pages. Thus, we use Hop-Distance Matrices (HDMs) while using low-rank matrix completion to predict missing hop-distances.

Even though there are theorems that give rise to interesting families of low-rank hop-distance matrices [143, 152], real-world graphs rarely, if ever, satisfy the assumptions of these theorems,

or many other similar theorems for both adjacency and hop-distance matrices. Accordingly, it is important to consider the *approximate low-rank properties* of hop-distance matrices. For example, one can consider common synthetic models that approximate the structure of real-world graphs such as *scale-free networks* or *power law graphs*, whose degree distributions follow, at least asymptotically, a power law. For example, one often considers graphs where the fraction of nodes N_k having k links goes like $N_k \propto k^{-\xi}$ for some parameter ξ [153, 154].

Accordingly, in Fig. 5.3, we show a comparison between the singular values of the adjacency matrix and the hop-distance matrix for two graphs, namely a synthetic power law graph based upon the Holme-Kim model [153] with 500 nodes and a subgraph of the real-world Gowalla social network from [155] with 2000 nodes. In addition, for the Holme-Kim synthetic data [153] we examine 100 Monte-Carlo runs to see the difference that the precise state of the random graphs makes. Note, in both cases the matrices are double-centered (as discussed in Section 5.5.7) and all singular-values are normalized relative to the size of the largest singular value (so all curves start on the left at 1). The singular-values of $\mathbf{H}$ decay much more quickly than the singular-values of $\mathbf{A}$, with the normalized 100th singular value of $\mathbf{H}$ in both cases being very close to 0, while the same for $\mathbf{A}$ are approximately 0.4 and 0.2 respectively.

Finally, we consider how the placement of anchor nodes affect the low-rank structure of the HDM. In Figure 5.4 we compare the low-rank structure of the HDM for the Holme-Kim [153] and Gowalla social network [155] between choosing a random set of anchors of $\mathbf{H}$ versus choosing a set of anchors based on different centrality measures. Somewhat surprisingly, the low-rank structure of $\mathbf{H}$ is not strongly affected by the choice of the sampling scheme. In particular, Figure 5.4 suggests that the accuracy of predictions from random anchors will be similar to that from anchors with high centrality (e.g., using nodes with a large number of neighbors as anchors) [143]. Note that the difference between the singular values of the hop-distance and adjacency matrices is much larger than the difference between the various sampling schemes. Of course, this does not mean that specially chosen anchors cannot change the low-rank structure of $\mathbf{H}$, but it does suggest that random anchors is a reasonable anchor selection for initial investigation.

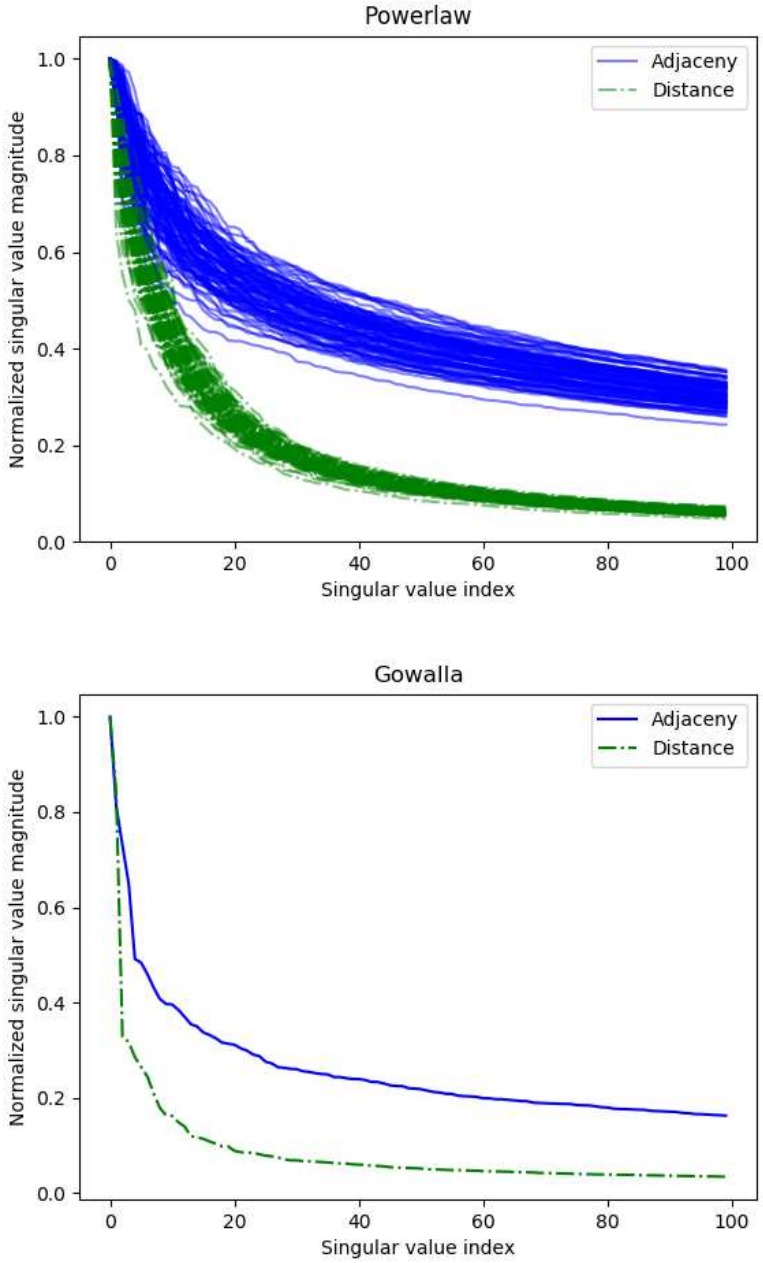


Figure 5.3: Comparison of the normalized singular values of adjacency matrices versus hop-distance matrices for synthetic and real-world power law graphs. Top figure is for a synthetic power law graph generated using the Holme-Kim model [153] with 500 nodes and bottom figure is for 2000 node subgraph of the real-world Gowalla social network from [155].

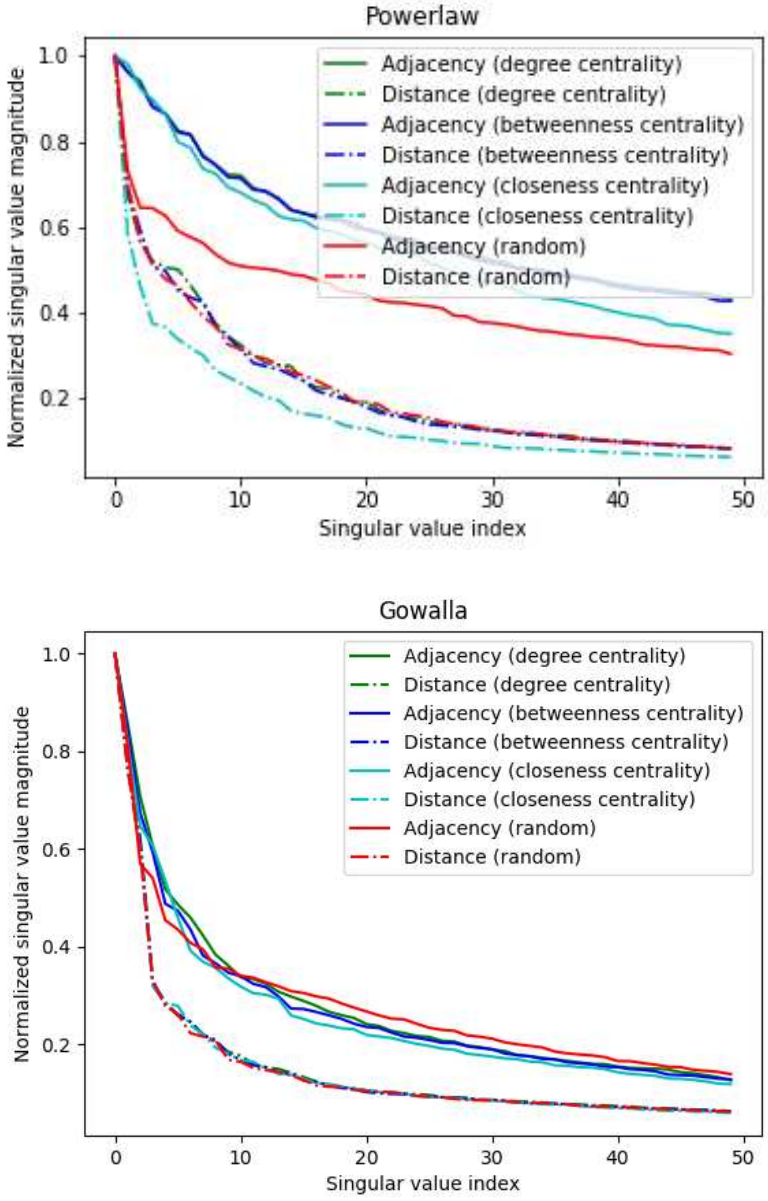


Figure 5.4: Comparison of the normalized singular values of adjacency matrices versus hop-distance matrices for a variety of anchor selection strategies for 100 anchor nodes. Top figure is for a synthetic power law graph generated using the Holme-Kim model [153] with 500 nodes. Bottom figure is for a subgraph of the real-world Gowalla social network from [155] with 2000 nodes.

5.5.6 Topology coordinates and Topology Preserving Maps

The mathematical foundation of our previous work in TPM generation from a VCS follows from the above formulation [82]. Consider the principal components of P given by,

$$\mathbf{P}_{\text{SVD}} = \mathbf{U}\Sigma \tag{5.7}$$

In the TC domain, each node in a 3D network is characterized by a triple of Cartesian coordinates $(x_T(i), y_T(i), z_T(i))$. Let $[\mathbf{X}_T, \mathbf{Y}_T, \mathbf{Z}_T]$ be the matrix of TCs for the entire set of nodes, i.e., the i-th row is the TCs of node i. Then from [82],

$$[X_T, Y_T, Z_T] = [\mathbf{P}_{\text{SVD}}^{(2)}, \mathbf{P}_{\text{SVD}}^{(3)}, \mathbf{P}_{\text{SVD}}^{(4)}] \tag{5.8}$$

where, $\mathbf{P}_{\text{SVD}}^{(j)}$ is the j-th column of $\mathbf{P}_{\text{SVD}}$. Note, in the derivation of TCs as presented in [82] the first singular vector $\mathbf{P}_{\text{SVD}}^{(1)}$ is *not used* in the representation. In Section 5.5.7, we will discuss the relationship between generating TCs without $\mathbf{P}_{\text{SVD}}^{(1)}$ and the idea of "double centering" [128].

The importance of TCs is that they capture the geometric features such as the shape and boundaries in spite of the fact that no Euclidean distance measurements are used. However, if some physical locations are known, then the TCs can be transferred to approximate physical coordinates as well [156].

5.5.7 Connections to Non-linear Dimension Reduction (NDR)

In this section, we look at some of the popular NDR techniques and how TPMs are related to them.

The "double centering" method of a given *squared* EDM $\mathbf{D}$ can be given by

$$\mathbf{S} = -\frac{1}{2}\left(\mathbf{D} - \frac{1}{N}\mathbb{1}\mathbb{1}^T\mathbf{D} - \frac{1}{N}\mathbf{D}\mathbb{1}\mathbb{1}^T + \frac{1}{N^2}\mathbb{1}\mathbb{1}^T\mathbf{D}\mathbb{1}\mathbb{1}^T\right) \tag{5.9}$$

where $\mathbb{1} \in \mathbb{R}^{n\times 1}$ is the vector all of whose entries are 1 [128]. In effect, $\frac{1}{N}\mathbb{1}\mathbb{1}^T\mathbf{D}$ is the matrix which contains all of the column averages of $\mathbf{D}$, $\frac{1}{N}\mathbf{D}\mathbb{1}\mathbb{1}^T$ is the matrix containing all of the row averages of $\mathbf{D}$, and $\frac{1}{N^2}\mathbb{1}\mathbb{1}^T\mathbf{D}\mathbb{1}\mathbb{1}^T$ is the matrix containing the average of all the entries of $\mathbf{D}$.

Double centering is a standard procedure for transforming a squared Euclidean-distance matrix into a Gram matrix to which SVD may be applied. Accordingly, applying double centering to

a low-rank hop-distance matrix produces a new space in which the Euclidean distances in that space approximate the original hop-distances. One may ignore the fact that the entries of the hop-distance matrix satisfy the triangle-inequality, and represent **D** directly by its SVD. However, double centering is more faithful to the intended structure of **H** and improves the interpretability of the resulting embedding.

MDS is one of the popular NDR techniques [128, 130, 131]. MDS performs low-dimensional embedding based on pairwise Euclidean distances between data points (nodes). PCA and MDS are equivalent when the inter-node distances are Euclidean. To obtain the low-dimensional embedding through MDS, one gets a gram matrix from the distance matrix by double centering, perform eigenvalue decomposition of this gram matrix, and then select top n eigenvectors in the descending order for n dimensional embedding.

Isomaps extend MDS by incorporating the geodesic distances [127,128]. The geodesic distance is defined here as the sum of edge weights along the shortest path between two nodes. The top n eigenvectors of the geodesic distance matrix represent the coordinates in the new n-dimensional Euclidean space.

TPMs are very similar to Isomaps in the sense that they also rely on the geodesic distances. However, we only consider unweighted edges, i.e., all the weights are equal to one and hence we simply compute the geodesic distances as the sum of hops. Note that a TPM is a low-rank approximation of a double centered **S** computed from **H** rather than a squared EDM **D**. In this sense, a TPM is analogous to the Multi-Dimensional Scaling (MDS) algorithm [128, 130, 131].

5.6 Approach

Proposed method for extracting network topology is as follows:

- Start with a set of geodesics. Two sampling schemes are considered: anchor-based VCs (to obtain **P** or a subset thereof) and random shortest paths (to obtain a subset of elements of **H**).
- Complete the matrix **P** or **H** using low-rank matrix completion.

- Evaluate the accuracy of the computed topology or layout. In the case of 2D and 3D sensor networks the accuracy of resulting topology preserving maps is used as the evaluation metric.

Here we describe the different elements of our approach in more detail.

5.6.1 Matrix completion

Prior work on TPM generation is based on the case where entire columns are taken from **H** and used to construct **P**. However, in the current work, we consider the more interesting, and practically important case, where each anchor node only has a *partial* set of measurements to the rest of the network. Accordingly, some entries in **P** are *not observed* and the matrix **P** is therefore incomplete. Predicting the unobserved entries in **P** can be phrased as a low-rank *matrix completion* problem. In particular, we have leveraged modern ideas in low-rank *matrix completion* which allow us to predict the missing entries in a partially observed HDM matrix. As our focus is on the development of theory and algorithms for treating HDMs, we refer the reader to [51, 87, 121–123, 129] for the details of matrix completion algorithms and this technique. In this document, we merely provide the reader with the intuition for such approaches in the context of predicting unobserved entries in HDMs.

Herein, one of our key assumptions is that the networks under analysis are low-rank, and therefore we have leveraged ideas from low-rank *matrix completion* [51, 123].

The main concept of such methods can be presented as the following optimization problem:

$$\hat{\mathbf{P}} = \arg\min_{\mathbf{P}_0} \rho(\mathbf{P}_0), \quad \text{s.t. } \mathcal{P}_\Omega(\mathbf{P}) = \mathcal{P}_\Omega(\mathbf{P}_0), \tag{5.10}$$

where ρ returns the rank of a matrix, $\mathbf{P}$ is a partially observed matrix, $\hat{\mathbf{P}}$ is the desired complete $\mathbf{P}$ matrix with all the missing entries filled in. In addition, Ω denotes a set of observed entries of the matrix $\mathbf{P}$ and $\mathcal{P}_\Omega$ denotes a projection of $\mathbf{P}$ onto Ω. Accordingly, the constraint above imposes that $\mathbf{P}_0 = \mathbf{P}$ for the observed entries of $\mathbf{P}_0$. The main idea is that we try to find a matrix $\mathbf{P}_0$ such that the rank of $\mathbf{P}_0$ is minimized while imposing that the observed entries are preserved in

the constructed matrix, i.e., $\mathcal{P}_\Omega(\mathbf{P}) = \mathcal{P}_\Omega(\mathbf{P}_0)$. Thus, the returned matrix $\mathbf{P}_0$ will be faithful to our measured hop-distances; but, $\mathbf{P}_0$ is free to take any values outside of Ω to minimize its rank. Unfortunately, Eq. (5.10) is an exponentially expensive optimization problem [51, 123] that can only be solved for trivial networks. However, the problem in Eq. (5.10) can be recast [51, 123] as a convex optimization problem as given below:

$$\hat{\mathbf{P}} = \arg\min_{\mathbf{P}_0} \|\mathbf{P}_0\|_*, \;\; \text{s.t. } \mathcal{P}_\Omega(\mathbf{P}) = \mathcal{P}_\Omega(\mathbf{P}_0), \tag{5.11}$$

which can be solved efficiently for millions of nodes on commodity computing hardware using splitting techniques and iterative matrix decomposition algorithms [121, 123]. In Eq. (5.11) we have exchanged the rank ρ for the *nuclear-norm* $\|\mathbf{P}_0\|_*$, which is the sum of the singular values of $\mathbf{P}_0$ which leads to a *convex* optimization problem [121, 123]. Even though a matrix of distances such as $\mathbf{P}$ should not have negative entries, the reconstructed matrix $\hat{\mathbf{P}}$ might have such entries. As LMC is not explicitly designed to handle such problems, we round-up all the negative values of $\hat{\mathbf{P}}$ to zero and positive values to the closest positive integer. There are many standard libraries for solving convex optimization problems such as Eq. (5.11). Herein, we have leveraged the library CVXPY [157] in the scripting language Python [158] to solve the problem in Eq. (5.11) using the pseudo-code as described in Algorithm 2. There have been several research solutions

Algorithm 2 Compute low-rank representation $\hat{\mathbf{P}}$ of the partially observed HDM $\mathbf{P}$.

Require: Partially complete hop-distance matrix $\mathbf{P}$.

1: **procedure** MATRIX COMPLETION
2: Compute recovered matrix $\hat{\mathbf{P}}$ using 5.11
3: Round-up all diagonal values of $\hat{\mathbf{P}}$ to 0 and off-diagonal values between 0 and 1 to 1
4: **end procedure**

on recovering missing entries of Euclidean distance matrices in undirected as well as directed graphs [159], however, to the best of our knowledge, recovering network data using only partial

geodesics from the test network is a novel technique and has not been done for undirected graphs such as scale-free networks.

5.6.2 Completion of partially observed hop-distance matrices

The double centering operation would seem to require a fully observed matrix **H**, negating our ability to analyze partially observed HDM matrices.

However, this difficulty in computing a "double centering" of a partially observed **P** can be overcome by way of the following equation, similar to Equation (5.9),

$$\mathbf{S}_{i,j} = -\frac{1}{2}\left(\mathbf{P}^2_{i,j} - \mu_j(\mathbf{P}^2) - \mu_i(\mathbf{P}^2) + \mu_{i,j}(\mathbf{P}^2)\right) \tag{5.12}$$

where $\mu_j(\mathbf{P})$ is the mean of the observed entries in the j-th column of **P**, $\mu_i(\mathbf{P})$ is the mean of the observed entries in the i-th row of **P**, and $\mu_{i,j}(\mathbf{P})$ is the mean of all of the entries in **H**. In effect, each entry of the double-centered matrix $S_{i,j}$ only depends on the square of the single entry $P_{i,j}$, along with *mean values* of the *rows* and *columns* of **P**. Accordingly, estimates of these mean values can be computed even for a partially observed matrix such as $\mathcal{P}_\Omega(\mathbf{P})$, by performing the required mean over just the observed entries of the appropriate column, row, or the entire matrix. Note that if a particular node has no measurements such a node cannot be predicted. Also, classically, Eq. (5.12) is defined for EDMs, and one might wonder whether it is applicable to HDMs. In fact, that is a salient point of our work, to find a space in which the Euclidean-distances and the hop-distances coincide.

In some sense, such restrictions on acceptable sampling schemes for **H** should not be surprising. For example, if a particular row of **P** (or **H**) contains no measurements then the distance from this node to any anchor is not known. In effect, nothing is known about this node and therefore no predictions can be made.

Identifying classes of acceptable and unacceptable sampling schemes is a topic for future research. However, such ideas are a close cousin of the incoherence requirements that arise in matrix

completion problems [51, 121, 122, 129]. Accordingly, drawing inspiration from that literature and for simplicity, we choose *random nodes as anchors* and measure their distances from *random nodes* in the network.

Our algorithm for recovery of complete $\mathbf{P}$ from partial entries and generating TCs from a partially observed HDM is described in Algorithm 3 below. Topology coordinates in [82] are given by 2nd and 3rd singular vectors in the case of 2D networks and 2nd, 3rd and 4th in the case if 3D networks as given in Equation 5.8. Thus as a comparison we use Algorithm 4, which carries out matrix completion directly on $\mathbf{P}$, and follows the approach in [82] to generate TPMs. Algorithm 3, based on double centering followed by the completion of the Grammian matrix $\mathbf{S}$, however, follows the approaches such as MDS for NDR in Euclidean spaces more closely.

Algorithm 3 Computing TPMs from a partially observed HDM matrix $\mathbf{P}$ via completion of Grammian matrix.

Require: Partially $\mathbf{P}$ of a graph $G = \{V, E\}$
Require: A target dimension k
1: **procedure** COMPLETE $\mathbf{S}$ FROM PARTIAL $\mathbf{P}$
2: Compute $\mathcal{P}_\Omega(\mathbf{S})$ from $\mathcal{P}_\Omega(\mathbf{P})$ using (5.12)
3: Compute approximate complete Grammian matrix $\mathbf{S}$ from $\mathcal{P}_\Omega(\mathbf{S})$ using (5.11)
4: Compute the SVD of the complete $\mathbf{S}$ as $\mathbf{S} = \mathbf{U}\Sigma\mathbf{V}^T$
5: **return** The first k columns of $\mathbf{U}\Sigma$ are the TCs
6: **end procedure**

Algorithm 4 Computing TPMs from a partially observed HDM matrix $\mathbf{P}$ via matrix completion.

Require: Partially $\mathbf{P}$ of a graph $G = \{V, E\}$
Require: A target dimension k
1: **procedure** COMPLETE $\mathbf{P}$ FROM PARTIAL $\mathbf{P}$
2: Compute approximate complete distance matrix $\mathbf{P}$ from $\mathcal{P}_\Omega(\mathbf{P})$ using (5.11)
3: Compute the SVD of the complete $\mathbf{P}$ as $\mathbf{P} = \mathbf{U}\Sigma\mathbf{V}^T$
4: **return** Columns 2 through $k + 1$ (i.e., the first column is excluded) are the TCs
5: **end procedure**

5.7 Results

Topological features in specially deployed networks such as IoTs, are restored from partial measurements of hop-distances in the network. Matrix Completion (MC) algorithm [160] is being leveraged to work with the partial distance entries in a graph to estimate the missing entries. Experiments are performed on sensor networks with various shapes and voids to depict practical conditions. We evaluate two networks in 2D and two networks in 3D space for various percentages of missing entries in the distance matrix. I have contributed to this work towards result computation and editing.

Here, we demonstrate the effectiveness of the proposed HDM based approach described in Algorithm 3 in constructing accurate topology maps from a small set of hop-distances among node pairs. In particular, we demonstrate how **P** or **H** can be recovered from a set of partial observations, and how topological coordinates that arise from the eigen-decomposition of **P** provide accurate recovery of relationships among network nodes. The evaluation is carried out for a set of 2D and 3D sensor networks. All the results have been averaged over 100 experimental iterations. As these results demonstrate, our methods provide surprisingly accurate predictions, even when only a tiny fraction of the network has been measured.

5.7.1 Recovery of networks embedded in 2D/3D spaces

Four networks representative of 2D and 3D sensor network deployments covering a range of shapes and sizes are used for the evaluation. They contain complex features such as convex and concave boundaries and voids.

- A concave 2D network with 550 nodes, the physical layout of which is shown in Figure 5.6(a) [82].
- A 2D circular network with multiple circular voids of 496 nodes as shown in Figure 5.7(a) [82].

- A 3D network, shown in Figure 5.8(a), consisting of 1640 nodes, which occupies a cube shaped volume with a hollow region in the shape of an hourglass devoid of nodes [85].

- A 3D surface network, shown in Figure 5.9(a), consisting of 1245 nodes, which is comprised of two hollow cylinders joined in a "T" configuration [82].

As an initial exploration of the applicability of the techniques we propose, we first examine the rank of the VC matrices (**P**) for each of our networks. Twenty random anchors were selected in each case, i.e., $M = 20$, which corresponds to approximately 3.6%, 4%, 1.2%, and 1.6% of the nodes, respectively, for each of our four networks. The singular values of the full VC matrices of the four networks are shown in Figure 5.5. If we were considering EDMs, then the rank of the first two networks would be 4 (since they are embedded in $\mathbb{R}^2$) and the rank of the second two networks would be 5 (since they are embedded in $\mathbb{R}^3$) [138, 140]. As seen in Figure 5.5, the rank of the HDMs is certainly higher than their embedding dimensions would indicate, if they were EDMs. This confirms the fact that the HDM of a network is fundamentally different from its EDM, even for cases where EDM exists, and thus one cannot rely on EDM based methods for operations related to connectivity. However, somewhat surprisingly, even though the four networks are quite different, all of their ranks are substantially smaller than 20, for the chosen random anchors. Our interest is in the recovery of topological information and geometric relationships such as the general shapes of boundaries, voids in the networks, and node neighborhood preservation. Thus, the question is whether such information is preserved and can be extracted from small numbers of anchors and partial observations of **P**.

Two-dimensional TPMs extracted using the full set of VCs following [82] are shown in Figure 5.6(b), Figure 5.7(b), Figure 5.8(b), and Figure 5.9(b), respectively, for the four networks. It is important to note that even the full set of VCs corresponding to 20 anchors, which corresponds to 20 random columns of **H**, contains only approximately 3.6%, 4%, 1.2%, and 1.6% elements of the corresponding HDM.

Next, we randomly discard 10%, 20%, 40%, and 60% respectively of this already small sample of the elements of **H**. The TPMs recovered using low-rank matrix completion followed by TPM

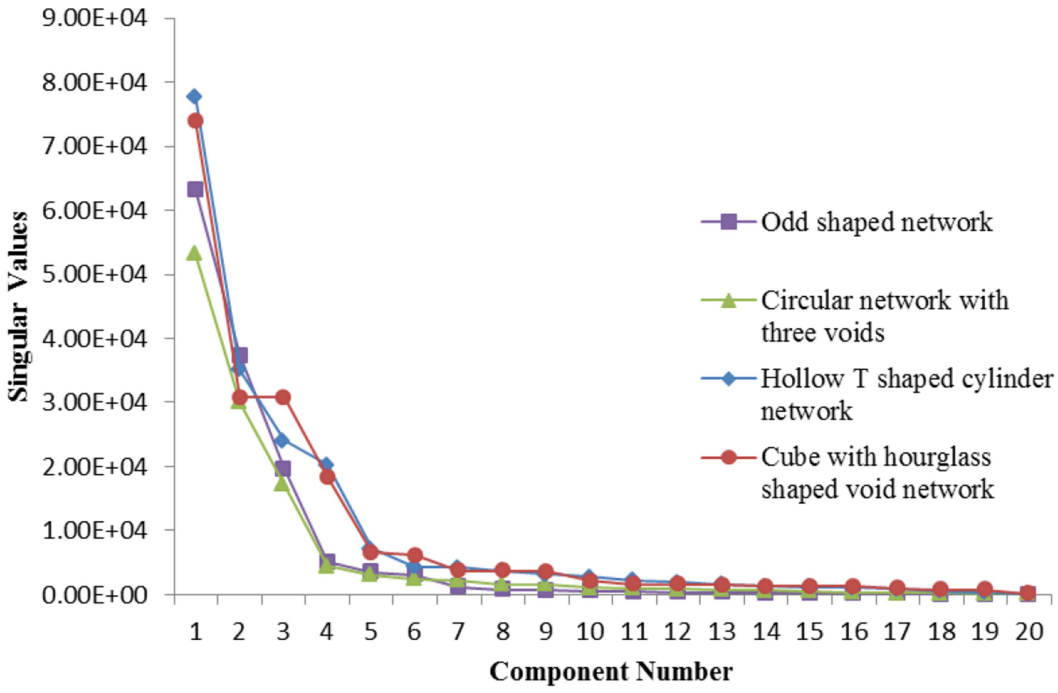


Figure 5.5: Singular values of the VC matrix for Circular Network, Odd-Shaped Network, Hollow T cylinder network, and the 3D network with void indicating the low-rankness of VCS data.

extraction are shown in Figure 5.6(c-f), Figure 5.7(c-f), Figure 5.8(c-f), and Figure 5.9(c-f) for these various sub-samplings. The results indicate that accurate TPMs of networks are obtainable with only a fraction of virtual coordinates. It is important to recognize that the goal of this work is not to necessarily recover the maps in subfigures (a) of Figures 5.6-5.9. Rather, we wish to recover subfigures (b) (the fully observed topology map) from a sparse set of hop-distance observations.

5.7.2 Accuracy of topology preservation

To precisely quantify the error in the estimated HDM due to missing VCs, we define the mean error E as follows:

$$E = \left[\sum_{i=1}^{N}\sum_{j=1}^{M}|d_{ij}(f) - d_{ij}(0)|\right] \Big/ \left[\sum_{i=1}^{N}\sum_{j=1}^{M} d_{ij}(0)\right], \tag{5.13}$$

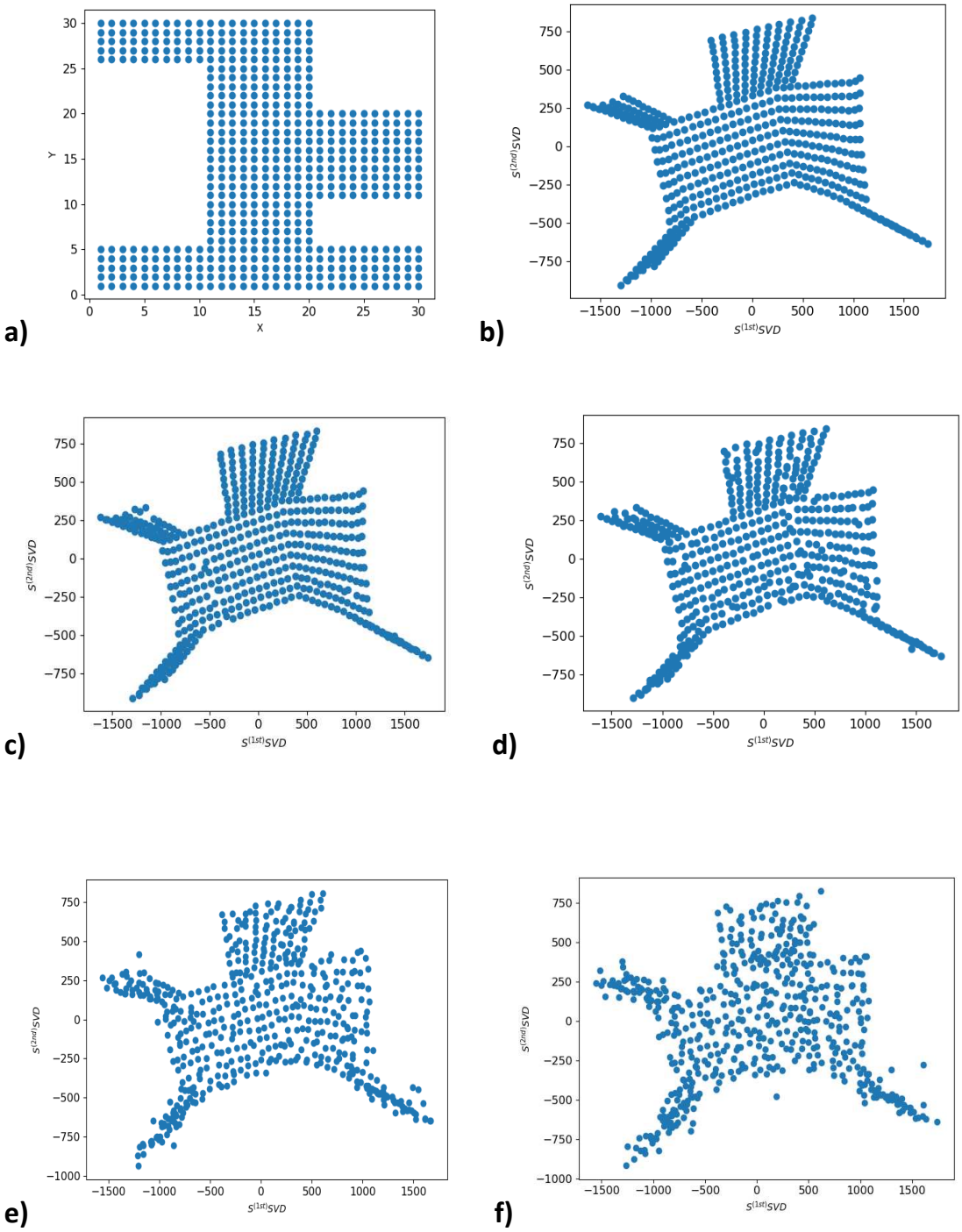


Figure 5.6: Concave network: (a) Original layout, and (b) TPM recovered from full set of VCs with 20 random anchors; Recovered TPM with (c) 10%, (d) 20%, (e) 40%, and (f) 60% of sampled coordinates randomly discarded.

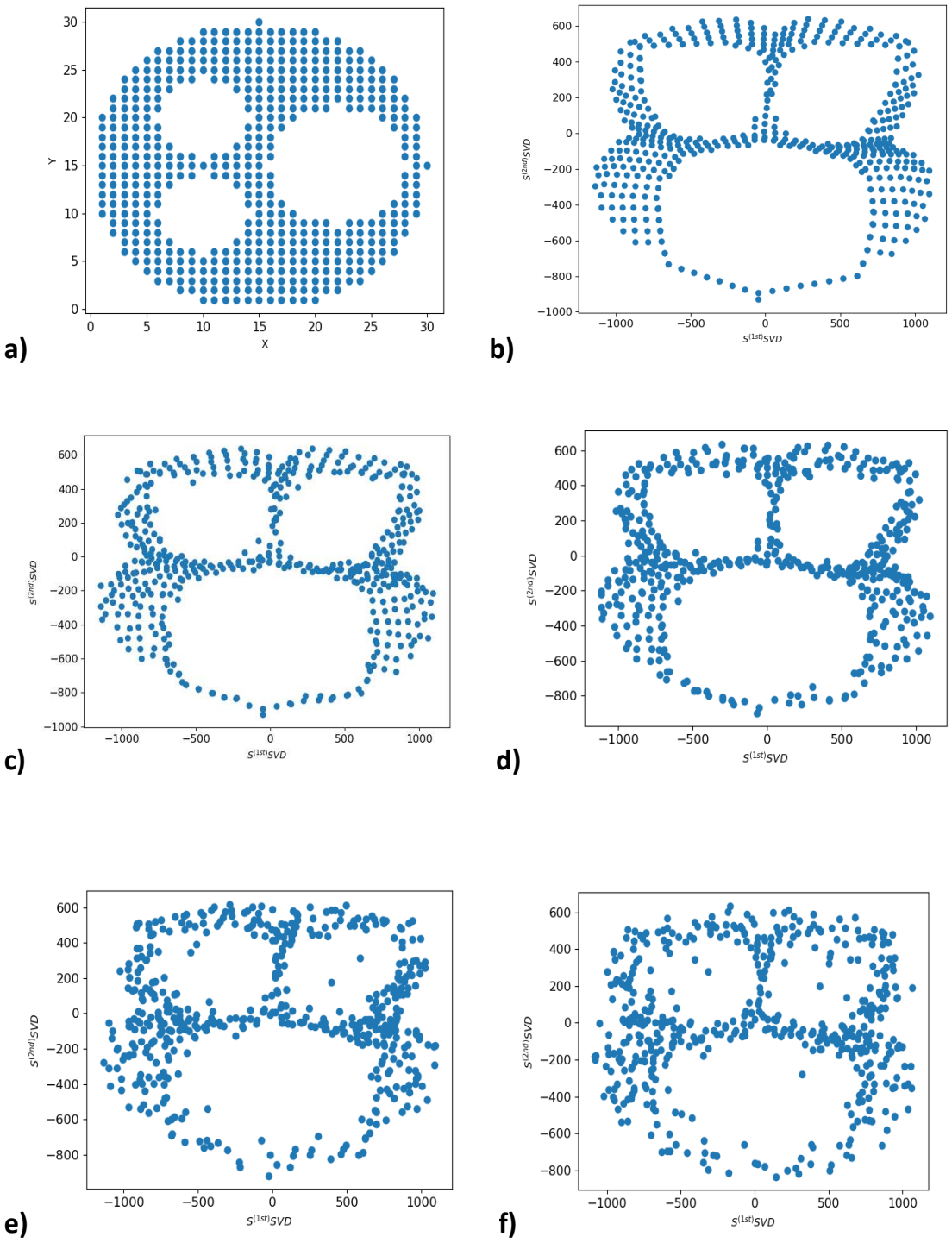


Figure 5.7: Circular network: (a) Original layout, and (b) TPM recovered from full set of VCs with 20 random anchors; Recovered TPM with (c) 10%, (d) 20%, (e) 40%, and (f) 60% of sampled coordinates randomly discarded.

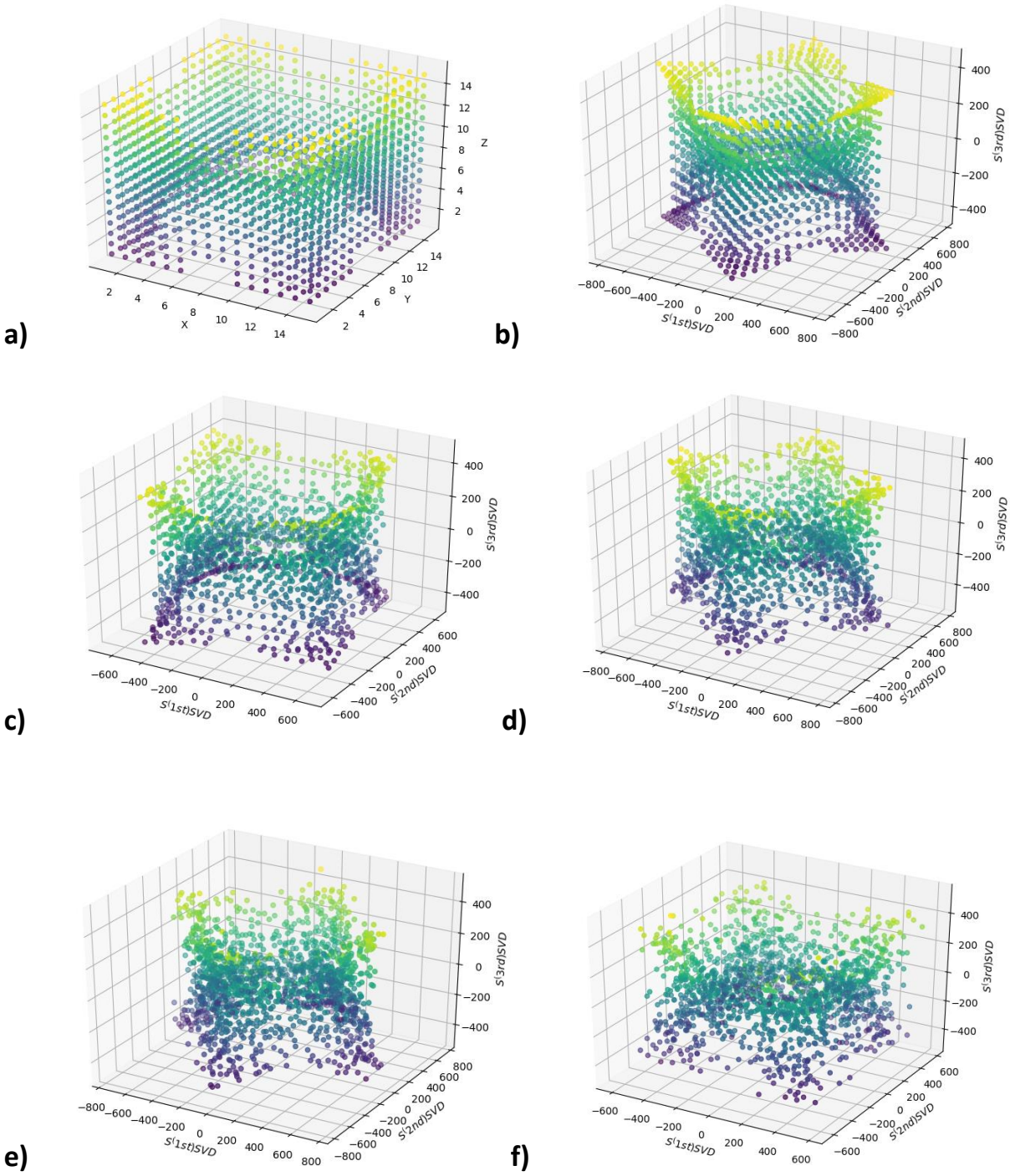


Figure 5.8: Cube with hourglass shaped void: (a) Original layout, and (b) TPM recovered from full set of VCs with 20 random anchors; Recovered TPM with (c) 10%, (d) 20%, (e) 40%, and (f) 60% of sampled coordinates randomly discarded.

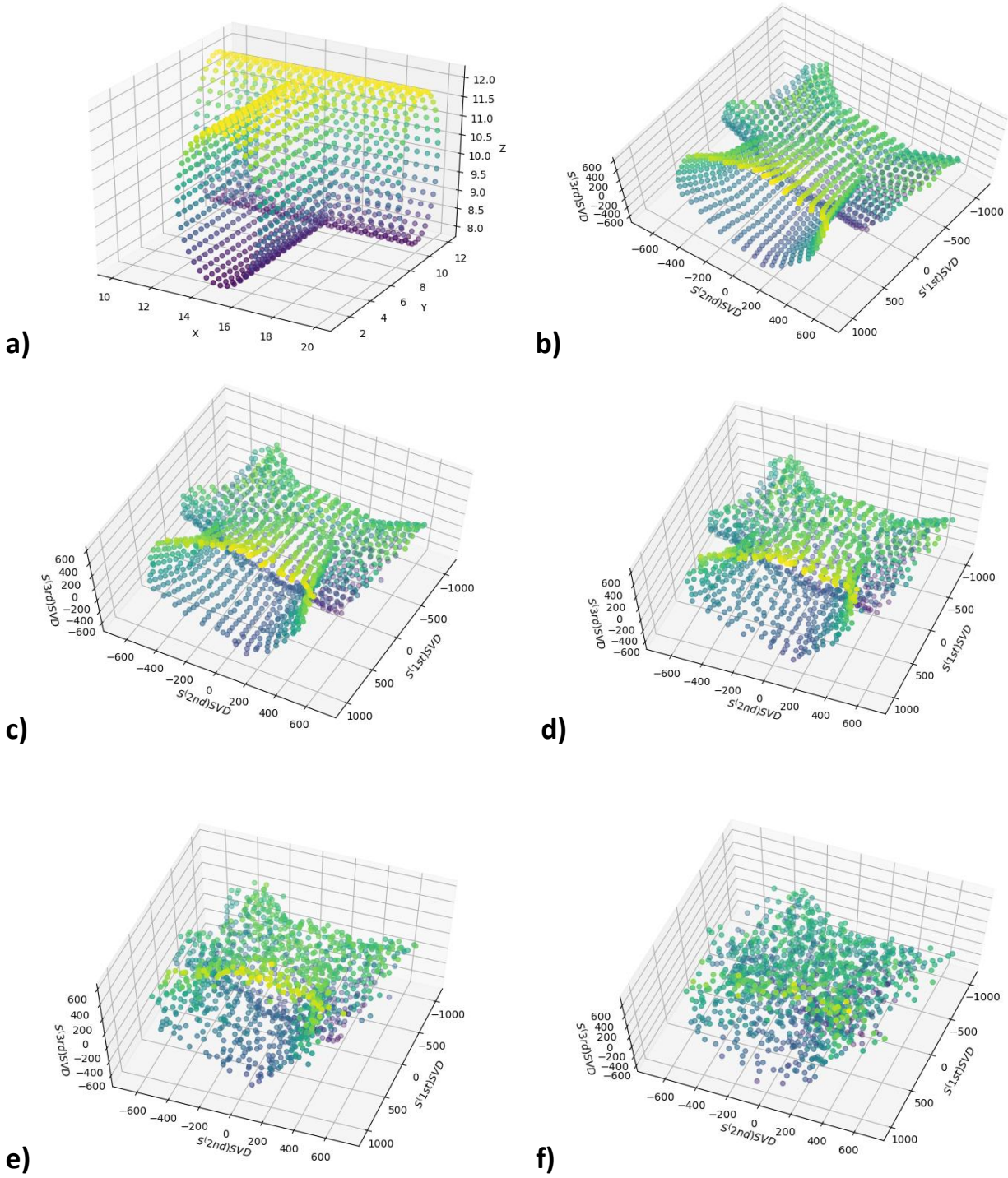


Figure 5.9: Hollow T shaped Cylinder: (a) Original layout, and (b) TPM recovered from full set of VCs with 20 random anchors; Recovered TPM with (c) 10%, (d) 20%, (e) 40%, and (f) 60% of sampled coordinates randomly discarded.

where, $d_{ij}(f)$ refers to the estimated hop-distance between nodes i and j when f fraction of random anchor coordinates are missing. The percentage mean error with percentage of missing VCs for the four networks are shown in Figure 5.10. It is important to note that even when mean error is high, as we later show in Section 5.7.3, local neighborhood and shape information is accurately preserved.

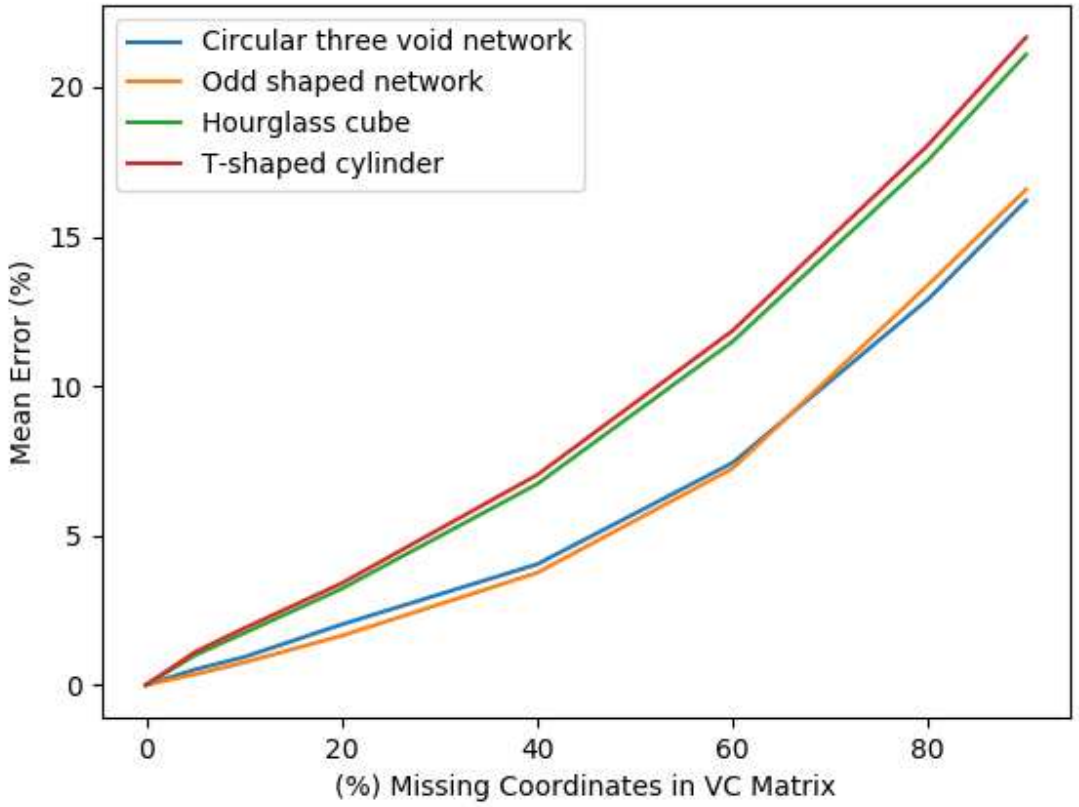


Figure 5.10: Percentage mean error $(E\%)$ versus the percentage of missing virtual coordinates $(f\%)$ for the four sensor networks. Number of anchors $M = 20$, which corresponds to 1-4% of the nodes depending on the network. Therefore $f = 80\%$ implies that the number of distance entries measured is in the range $0.02 - 0.08\%$.

5.7.3 Accuracy of Topology Preserving Maps

The accuracy of neighborhood preservation of reconstructed topology maps of 2D networks is evaluated below using the topology preservation error E_{TP}, which captures the degree to which the neighborhood relationships are altered. We provide only a brief explanation of E_{TP} and refer the reader to [82] for its precise definition. Consider the network in Figure 5.6(a). The network is scanned along with the set of lines in the horizontal direction ($\vec{H}$) and the vertical direction ($\vec{V}$). Let α and β denote the sets of lines in $\vec{H}$ and $\vec{V}$ directions respectively. Consider one such line which contains the ordered set of some m nodes $\{n_1, ...n_m\}$, which we call the original placement. Now consider the projection of this set of nodes in a TPM (e.g., Figure 5.6(b)) on the corresponding axis. An error indicator function $I_{i,j}$ is defined by comparing the order of projected nodes with the original placement as:

$$I_{i,j} = \begin{cases} 1, & \text{nodes } i \text{ and } j \text{ are out of order,} \\ 0, & \text{nodes } i \text{ and } j \text{ are in the same order or } i = j. \end{cases} \tag{5.14}$$

For the line under consideration, the neighborhood preservation error is quantified by $\sum_{\forall i,j}(I_{i,j})/{}^{m}C_2$, where ${}^{m}C_2$ is the number of 2 combinations in m (m choose 2). However, in the case of TPM, we are interested in the overall neighborhood preservation error E_{TP} over the set of lines in $\vec{H}$ and $\vec{V}$ directions. Let N_h and N_v be the number of nodes in a given horizontal and vertical line respectively. Thus, the 2D TPM error is given by [82]:

$$E_{TP} = \left[\sum_{\alpha}\sum_{\forall i,j}(I_{i,j}) + \sum_{\beta}\sum_{\forall i,j}(I_{i,j})\right] \Bigg/ \left[\sum_{\alpha} {}^{N_h}C_2 + \sum_{\beta} {}^{N_v}C_2\right] \tag{5.15}$$

Next, we evaluate the effectiveness of matrix completion with HDMs for TPM generation using the approaches in Algorithm 3 and Algorithm 4. E_{TP} (as a percentage) for the 2D networks for these two cases provided in Table 5.1 show that with merely 20 random anchors, which corresponds to only 4% of the nodes, networks can be recovered with an error less than 32% even after deleting

80% of the entries. The TPM error for 2D networks is under 15% for 40% deletion of entries. E_{TP} for 3D networks, hourglass, and T-shaped pipe, are presented in Table 5.1.

We extended the Topology Preserving error to 3D networks where we consider all the planes through the network parallel to X-Y, Y-Z and X-Z planes, evaluate $E_{TP}\%$ using Eq. (5.15) on each plane, and report their average [161]. E_{TP} for 3D networks, hourglass, and T-shaped pipe, are presented in Table 5.1. We select a sample of merely 20 random anchors, which corresponds to less than 2% of nodes in the given 3D networks. Error in T-pipe network is higher than in Hourglass network due to the sparse placement of nodes in its topology.

Table 5.1: Topology preservation errors ($E_{TP}\%$) for 2D and 3D networks: Circular, Concave, Hourglass, and T-pipe networks with 10% to 80% of sampled coordinates randomly discarded. Subscript **S** denotes results with completion of Grammian matrix and taking 1st and 2nd singular vectors (Algorithm 3), while subscript **P** denotes completion of the **P** matrix and taking 2nd and 3rd singular vectors (Algorithm 4).

Deletions(%)	0%	10%	20%	40%	60%	80%
$Circular_S$	8.83	1.88	3.47	6.71	12.91	19.16
$Circular_P$	7.94	1.95	3.07	9.35	15.50	20.87
$Concave_S$	6.59	0.81	1.45	3.99	9.83	16.85
$Concave_P$	11.88	3.57	7.17	10.37	16.06	22.22
$Hourglass_S$	7.60	0.74	2.74	5.76	11.34	17.44
$Hourglass_P$	8.09	2.21	3.16	6.92	12.14	15.52
$T-pipe_S$	8.57	1.56	3.82	6.39	12.43	18.67
$T-pipe_P$	15.47	8.16	11.68	17.86	23.18	27.34

The results show that the presented technique can be used to recover 3D networks with an error of less than 32% even after deleting 80% of the entries from the small sample of network measurements.

5.8 Conclusion

This paper addresses the problem of recovering network features from a small set of hop-based graph geodesics. For networks deployed on 2D surfaces and 3D spaces the geometric and physical layout features are of importance. The approach starts with anchor-based VCs but, unlike prior techniques that required the entire VC set, the proposed approach requires only a fraction of the

measured VCs to recover accurate topology preserving maps. Our technique is based on the theory of low-rank matrix completion that reconstructs missing VCs, the result of which is used to recover layout maps using topology preserving map generation techniques.

With random anchor selection, the VC matrix (**P**) is equivalent to a small set ($< 10\%$) of random columns of an HDM. Thus, the partially complete **P** matrix where a large set of random VC entries are missing is equivalent to an incomplete HDM with only a very small number of entries. We also demonstrated that the HDMs of many real-world networks are low-rank. Therefore, the approach presented here provides a foundation for designing novel graph sampling techniques that allow the capture of complex real-world networks with a small number of measurements. The ability to accurately estimate missing network topology information from sparse distance measurements as presented here not only allow the reduction of cost (communication, power, etc.) of VC generation but, more importantly, open the possibility of using virtual-coordinate based and hop-distance based techniques for large networks and even those involving soft-state systems, where some coordinate values may be allowed to expire, thus allowing for more resilient network operations.

Beyond results for physically embedded networks, this work can also be extended to and tested on networks that do not follow Euclidean distances, such as social networks where the distance between two nodes cannot be measured in L_2 norm. Hop-distances play a crucial role in representing node pair distances in such cases. The ability to make accurate predictions in a large social network also from a small set of random geodesic measurements could be tested in the future. It is interesting to observe that there are many distances and similarities that could also be considered using this type of analysis. For example, as was pointed out, PageRank [162] can be used as a similarity measure by computing the probability that a random talk from one node arrives at another node in a given number of steps. Studying the low-rank properties of such similarity measures would be a rich direction for future research.

Chapter 6

Predicting Missing Node-pair Distances in Social Networks Using Partial Measurements

We extend the application of Low-rank Matrix Completion [160] to real-world social networks in order to estimate the missing entries. The proposed method is a significant generalization that allows the topology to be extracted from a random set of graph shortest paths, making it applicable in contexts such as social networks where VC generation may not be possible.

Though the same estimation method of Low-rank Matrix Completion is used, it cannot be directly mapped to social networks from sensor networks for the following reasons: a) Social networks tend to have different degree distribution unlike sensor networks for which the degree distribution is almost uniform b) Social networks do not relate with spatial distances. Such networks are represented and measured in terms of hop-distances for connectivity and network structure whereas sensor networks are analyzed to extract their geometric properties.

Accordingly, in this chapter, we employ Low-rank Matrix Completion to estimate missing node-pair distances in undirected and evaluate the system performance (Section 6.4) as well as directed (Section 6.9) social networks. We test the prediction scheme on several real-world social networks and also present bounds for missing node-pair distances in undirected networks.

Thus, we use Matrix Completing [160] in hop-distance domain while following the constraints of triangle inequality so as to maintain the true nature of a distance matrix. This is done by performing Matrix Completion under the constraints of bounds computed in Chapter 4, Section 4.5. Experiments are performed on sampled Facebook, Email, and Collaboration networks.

6.1 Introduction

With the proliferation of applications allowing people to instantly interact with each other over distances, social networks have become a valuable source of information for commerce, defense,

and scientific research. Capturing social relationships, mining information from social networks, and use of social relationships for purposes such as marketing have become widespread. Many of the social networks of interest are massive and complex in connectivity. Well known examples include Facebook with over a billion users, Twitter with around 300 million users, citation networks, and collaboration networks. Carrying out analytics and deep learning over these social networks requires access to enormous computational, storage, and communication resources. In certain instances, this may limit the process of operating on a manageable subset of the original network. Efficient mining of information on such a large and complex network is thus a significant challenge. While information about these networks may reside on the worldwide distributed computing and communication infrastructure, many of the operations of interest such as mining information, characterization of nodes, and topology, etc., cannot rely on exhaustive access to the nodes and links.

Novel scalable approaches are required to move beyond brute-force capture, storage, and extraction strategies. Such strategies can exploit the fact that these network structures are not purely random nor densely connected with respect to the network size. For example, the Facebook network on a global scale affirms the principle of “six degrees of separation" with almost 99.6% of all pairs of users within six hops [163], with an average path length of 4.7. Facebook is almost fully connected with 99.91% of individuals belonging to a single large connected component. An interesting thing to note is that the Facebook graph is sparse, but graph neighborhoods possess dense structures [163]. Analysis of Flickr and Yahoo! 360 shows that the average diameters of these networks slightly exceed the six degrees of separation, however, with increasing edge density the diameter starts to fall [164].

The connectivity and topology of a network often are the most valuable information in a social network. Most of the relationship information of interest is embedded therein. Applications ranging from advertising to counter-terrorism are all supported by a good understanding of the network connectivity and topology. They also play a major role in determining the speed and the extent of information being spread via networks. For example, information could spread rapidly among a

well-connected subset of nodes, but at the same time, it subjects these nodes to information clutter resulting in their potential to take profitable actions. However, our understanding is often impeded due to the overwhelming complexity of these networks, the inability to enumerate all the links, and lack of methods that can fully exploit the sparsity of network structure for efficiency. Here, we address the capture, representation, and reconstruction of networks with a relatively small set of measurements, thus paving the way for dealing with large-scale networks with significantly less computation, communication, and storage resources compared to dealing with full network topology.

Social networks such as Facebook, Instagram, and Orkut represent large-scale networks. Controlled and focused crawling [165, 166] are widely used approaches for data collection in such networks for acquisition of data regarding hop-distances, links, etc. As some profiles (nodes) are publicly accessible and some are not [35], we cannot gather data for all the nodes while crawling. In such cases, the hop-distances acquired, or the links observed are only partial entries in the complete distance or adjacency matrix. Also, as proved in [20], missing data can have a significant impact on the structural properties of a social network.

The main contribution of this section is a novel technique that exploits the sparsity inherent in social networks. This allows us to use a small number of distance measurements to capture and effectively reconstruct network topology, thus significantly decreasing the complexity associated with exhaustive network sampling, storage, and topology extraction methods. A fundamental question that we address is "What information about the network topology is encoded in a small subset of elements of its distance matrix **D**?". Also, for a vast variety of networks, e.g., social and computer networks, *the concept of Euclidean distances is not even applicable* (e.g., what is the "Euclidean distance" between two people in a social network?). We extend the matrix completion technique to hop-distance matrices and also compute bounds for the unknown entries. The effectiveness of our method is validated using three sub-networks sampled from different social networks (Facebook: 744 nodes, Arxiv: 4158 nodes, Enron e-mail: 4989 nodes).

The results presented are applicable in multiple contexts.

1. First, the derivation of upper and lower bounds presented for missing entries in a distance matrix can be applied to efficiently predict missing data which was inaccessible during network sampling.

2. Second, the proposed approach allows for the reconstruction of the graph from a small subset of measurements using matrix completion even in hop-distance domain, which is invaluable when using the complete set of measurements is impossible.

3. Third, the approach may be viewed as a compression strategy for massive networks, where the network topology is described using a small set of values rather than the full set of connectivity.

4. Fourth, many of the mining operations may now be performed on this compressed representation. Our results show that a significant degree of compression may be achieved with no or little loss of information.

As we test our proposed reconstruction technique on both undirected and directed networks, we divide the theory, results and discussion in separate sections. This paper is structured as follows: Sections 6.2 and 6.8 defines our notation and introduces the context and assumptions, two topology capturing techniques, virtual coordinates, and random path measurements for undirected and directed networks. In Section 6.3, we discusses our topology reconstruction method. Sections 6.4 and 6.9 provide results with undirected as well as directed networks respectively. Lastly, Sections 6.5 and 6.10 summarizes our work for undirceted and directed networks.

6.2 Theory

6.2.1 Context and Notation

This paper presents a scalable approach to capture, store, and extract information in large-scale social networks. Our focus here is on undirected networks. Consider a social network with N nodes, which we represent by an undirected graph $G = \{V, E\}$, where V is a set of vertices

(nodes) and E is a set of edges (unordered pairs of nodes corresponding to links in the network). An unweighted graph, often used to represent a social network, can be completely characterized by its *adjacency matrix* **A** [143], defined as

$$\mathbf{A} = [a_{ij} | a_{ij} = 1; \text{if } (i,j) \in E, 0 \text{ otherwise }] .$$

We summarize all the notational conventions that we used in the paper in table 7.1 for easy reference.

Table 6.1: Notation

Notation	Description
$\mathcal{N}$	Set of nodes
$N = \|\mathcal{N}\|$	Number of nodes
$n_i \in \mathcal{N}$	i-th node (current node)
$\mathcal{A} \in \mathcal{N}$	Set of anchor nodes
$M = \|\mathcal{A}\|$	Number of anchors
$A_i \in \mathcal{A}, i = 1, \ldots, M$	i-th anchor
h_{ij}	Minimum hop-distance between node n_i and n_j
$\mathbf{D}$	Hop-distance matrix
$\mathbf{P}_{(i)} = [h_{iA_1}, h_{iA_2}, \ldots, h_{iA_M}]$	Virtual Coordinates (VC) of node n_i (we define VC in detail in 6.2.2)
$\mathbf{P}_{N\times M} = [h_{iA_j}]_{ij};\ i = 1, \ldots, N,$ $j = 1, \ldots, M$	VC set for entire network
$\mathbf{Q}_{R\times M}; R \ll N$	VCs of a subset of nodes
f	Fraction of missing anchors with missing coordinates
λ_i	i-th singular value
k	Neighborhood size
C	Average clustering coefficient
ξ	Average node degree

Without loss of generality, in this paper we will consider only connected networks. In the case where the network consists of multiple disconnected components the approach is applicable for each connected component, but the results for different components can easily be combined in a separate step. For connected networks, instead of **A**, we consider the *hop-distance* matrix $\mathbf{D} \in \{\mathbb{Z}_0^+\}^{N\times N}$. We denote the ij^{th} element of **D** by h_{ij}, where

$$h_{ij} = \text{the length of the shortest path (in hops) from node } i \text{ to node } j.$$

The two representations **A** and **D** contain complete information about the topology of a network and given one, the other can be obtained. Thus, the capture, storage, and extraction of information of a social network typically involves manipulation of this huge matrix in one form or another. **A** is a sparse binary matrix with non-zero elements corresponding to different links. However, often it is not possible to capture the complete direct connectivity information to construct the adjacency matrix, i.e., links between each set of neighbors of each node in a network, owing to reasons such as privacy and secrecy. Furthermore, the number of possible graphs for an **A** with missing entries increases exponentially with the number of such missing entries as each missing entry is independent of the others (without additional assumptions) and can have two values (0 or 1). **D**, in comparison, is a more structured matrix with row (column) i containing all the integers in the range $[0, r_i]$ where r_i is the maximum distance to a node from node i. A random set of shortest path length measurements among node pairs corresponds to a random set of elements in **D**. Oftentimes, sampled network data does not guarantee preservation of original network characteristics due to loss of information through unmeasured parameters [20]. In this article, we address this problem. The proposed approach is based on exploiting the structure and properties of **D** to recover the topology and connectivity of the original network from a small set of such (random or strategically sampled) partial measurements.

Our work is aimed at networks where; (a) it is not possible to obtain the complete information, which may be due to complexities and restrictions associated with measurements or the size of the graph; or (b) the size of the graph is so large that it is only feasible to manipulate a small subset of information appropriately sampled.

The term “distance matrix” occurs more commonly with respect to Euclidean spaces, in which the ij^{th} element corresponds to the Euclidean distance, e.g., between two vectors or two geometric points. As it is used in many contexts, such as dissimilarity of data points or images, in geometry, and occasionally to approximate network physical layouts, there exists a large body of literature related to its estimation [167, 168]. However, the Euclidean distance matrix is substantially different from our use of “distance matrix" of hop-distances in graph topology, in which the notion of an Eu-

clidean distance does not exist. Distance-free localization approaches in sensor networks which are aimed at the recovery of Euclidean coordinates often start with hop-distances but then use different techniques, such as geodesic distances, to obtain approximations to Euclidean coordinates. In addition, we note that such sensor network topologies are much simpler and constrained compared to social network topologies since the sensor networks are embedded in 2D and 3D physical spaces, that naturally satisfy Euclidean geometric properties, with node neighborhood limited by wireless transmission. Topology coordinate space [82] is a Euclidean space derived purely from the hop-distances by using PCA and serves as an alternative for physical coordinates for sensor network operations. This work uses the concept of Virtual Coordinates Systems (VCS) [78,169] as the starting point, which essentially corresponds to a subset of columns of the hop-based distance matrix, which we also rely on for the current work. The low-rank matrix completion technique has been used successfully for many years in completion of Euclidean distance Matrices (EDM) [138, 140] as for a EDM E, $rank(E) \leq k + 2$ where k is the dimension of the embedding space whose pairwise distances comprise E [138]. However, hop-distance matrices do not qualify for low-rank structure through this property.

To our knowledge, the problem of recovering the hop-distance matrix capturing the network topology of a graph from a subset of its elements has not been addressed before.

As we will show in Section 7.4 for social networks and in [82, 85] for sensor networks, the hop-distance matrices $\mathbf{D}$ for many interesting and realistic networks are, somewhat surprisingly, *approximately low-rank*. It is this empirical observation that inspires our work. Singular Value Decomposition (SVD) [145] is used to analyze the low-rankness of a matrix, in Principal Component Analysis (PCA) [146] and dimensionality reduction. SVD of a matrix $\mathbf{P} \in \mathbb{R}^{N \times M}$ can be written as

$$\mathbf{P} = \mathbf{U}\Sigma\mathbf{V}^T, \tag{6.1}$$

where $\mathbf{U} \in \mathbb{R}^{N \times \min(N,M)}$, $\Sigma \in \mathbb{R}^{\min(N,M) \times \min(N,M)}$, $\mathbf{V} \in \mathbb{R}^{M \times \min(N,M)}$, and $\mathbf{U}\mathbf{U}^T = \mathbf{V}\mathbf{V}^T = I$.

The diagonal entries of Σ are called the singular values of $\mathbf{P}$. The rank of $\mathbf{P}$ is the number of singular values not close to zero. Given a low-rank matrix $\mathbf{P}$, the entire matrix can be approximated by using just a few columns of $\mathbf{U}$ and $\mathbf{V}$, and just a few entries of Σ. Low-rankness also allows the matrix to be completed from an appropriate subset of its entries. There is a rich literature related to techniques for analyzing such low-rank structures, including spectral graph theory [170–173] that relates the properties of a graph to the characteristic polynomial, eigenvalues/singularvalues, and eigenvectors/singularvectors of these matrices.

6.2.2 Topology Capture Techniques

There is a rich literature that suggests ways to capture information from social networks with web crawlers [35, 174–176]. Some of the common graph sampling techniques on social network employ either graph traversal method or random walk methods. The common graph traversal methods are Breadth First Search (BFS) or Depth First Search (DFS). However, it may lead to bias towards higher degree nodes. The latest work on random walk-based graph sampling addresses this issue of bias [174]. Owing to the massive size of such graphs, significant computing, and memory resources are needed to capture data. Attention is to be paid while choosing the origin node that initiates the graph sampling to make sure that the right population is sampled. The Algorithm 7 gives a generic method to capture the connectivity from a social network. The current research work is on undirected graphs and hence, the reciprocity of nodes is assumed.

Our approach to capture the network topology is based on a small number of distance measurements among node pairs. We demonstrate the recovery of network topology using two different types of measurement sets (or connectivity information subsets), namely, the distances from a small set of anchor nodes, and a random set of path measurements. The goal is to recover the complete matrix that we start from, i.e., either $N \times M$ subset of the distance matrix or the complete $N \times N$ distance matrix, while the complete distance matrix $\mathbf{D}$ can be given as

$$\mathbf{D} = \begin{bmatrix} h_{11} & \dots & h_{1M} & \dots & h_{1N} \\ \vdots & \ddots & \vdots & & \vdots \\ h_{M1} & \dots & h_{MM} & \dots & h_{MN} \\ \vdots & & \vdots & \ddots & \vdots \\ h_{N1} & \dots & h_{NM} & \dots & h_{NN} \end{bmatrix} \tag{6.2}$$

where h_{ij} is the shortest hop-distance between node i and node j. The two measurement sets that we use correspond to a small subset of elements of the distance matrix, and thus it should be noted that our approach is applicable for many other measurement sets that provide a subset of elements of **D**.

6.2.3 Virtual Coordinates (VCs)

The first measurement set, commonly known as anchor-based VC set [82], uses M out of N total nodes that serve as anchors while $M \ll N$. These M nodes may be self-selected random nodes, e.g., each node becomes an anchor with some probability. This is the strategy used for results in this paper. Alternatively, some type of strategy, e.g., based on connectivity or ease of measurement, may be used. Depending on the type of network, it may even be possible to select anchors centrally and inform those nodes. Each node is characterized with respect to its distance to each of these M anchors. Thus, our measurement set consists of the $N \times M$ sub-matrix of **D** indicated within parenthesis of Eq. (6.2), which is commonly known as the VC matrix.

As M nodes are chosen as anchors, each node is characterized by a VC vector of length M. Thus our measurement set is the $N \times M$ matrix **P** corresponding to the subset of columns of the distance matrix **D**. It is desirable to have only a small subset of nodes as anchors, i.e., $M \ll N$.

In particular, one can consider **P** as a *sub-matrix* of the full matrix

$$\mathbf{D} = \begin{bmatrix} \mathbf{A} & \mathbf{B}^T \\ \mathbf{B} & \mathbf{C} \end{bmatrix}, \tag{6.3}$$

then, $\mathbf{A} \in \mathbb{N}^{M\times M}$ contains the hop-distances between the M anchors and themselves, $\mathbf{B} \in \mathbb{N}^{(N-M)\times M}$ contains the hop-distances between the M anchors and the $(N-M)$ non-anchor nodes, and $\mathbf{C} \in \mathbb{N}^{(N-M)\times(N-M)}$ contains the hop-distances between the $(N-M)$ non-anchor nodes and themselves. In this way, $\mathbf{P}$ can equivalently be written as

$$\mathbf{P} = \begin{bmatrix} \mathbf{A} \\ \mathbf{B} \end{bmatrix}. \tag{6.4}$$

An anchor-based method is especially relevant for social networks where messages generated by a small set of prominent nodes are forwarded to all the nodes in the network. One possibility is to select such prominent nodes as anchors and keep track of the number of hops as their messages propagate to other nodes.

Algorithm 5 Generation of VC from anchors

```
procedure ANCHOR SELECTION
    M anchors are chosen randomly
end procedure

procedure AS AN ANCHOR NODE
    for all Random nodes M do
        Send broadcast message to its neighbors
        with a TTL of t.
    end for
end procedure

procedure AS AN INDIVIDUAL NODE
    if t > 0 then
        if h_{A_i} is minimum yet then
            Record the hop-distance from respective anchors
        else
            Forward the message to neighbor with
            t decreased by one.
        end if
    else
        Stop forwarding.
    end if
end procedure
```

Algorithm 5 describes steps for generating VCs from anchors at each node in the network. Each node knows its VC vector from all the chosen anchors. This approach may be implemented on a network, e.g., Facebook by broadcasting a message to friends and in turn encouraging them to forward the message. Not all of them would be interested to cooperate and the resulting measure will be a clear subset of the VC matrix. A Time-To-Live (TTL), t, should be set to limit the messages in the network. As mentioned earlier, social networks classically abide by the "six degrees of separation" rule [177], thus, setting the hop count to an integer equal to or slightly higher than 6 would capture all the information needed.

6.2.4 Random Path Measurements

For a social network, if the VC matrix is not available, we provide an alternative to capture the social network topology. Thus, the second measurement set corresponds to the shortest path distances between random node pairs. We consider random node pairs, in which case the measurement set consists of a small subset of random elements of **D**. This, in a certain sense, is equivalent to having an incomplete version of the virtual coordinate matrix **P** where many of the elements are unknown. These random samples of graphs can be obtained by using web crawlers or the data captured from the anchor based method can be reorganized to get it as a subsample of distance matrix **D**.

To summarize, we capture the network topology using either a) the set of distances from each node to a small subset of nodes, i.e., anchors, or b) the set of shortest distances among randomly selected node pairs.

6.3 Method: Topology Reconstruction

Irrespective of the capture technique used, our measurement set consists of a small set of elements of the distance matrix. Results presented later indicate that the distance matrices of large social networks, in general, are low-rank. Thus, we use results from low-rank MC to construct the network topology. Next, we provide details of the low-rank MC algorithm essential for its appli-

cation on social networks. For more information, one can refer to [87]. We have formulated this from ideas in low-rank MC [51, 121, 122, 129, 178]. The captured connectivity information forms a subset of elements in **D** for random measurements, and in **P** for a VC set. The low-rank MC can be explained as follows:

$$\begin{aligned} \mathbf{L} &= \arg\min_{\mathbf{L}_0} \rho(\mathbf{L}_0), \\ \text{s.t. } &\mathcal{P}_\Omega(\mathbf{D}) = \mathcal{P}_\Omega(\mathbf{L}_0), \end{aligned} \tag{6.5}$$

where ρ is the rank operator and Ω is the set of observed entries in **P**, so that $\mathcal{P}_\Omega$ is the known subset of elements of **P**. We are trying to find a matrix **L** which predicts the unknown values in such a way that, $\rho(L)$ is minimized and it meets the constraints of existing hop-distance values. Eq. (7.3) is an NP-hard[1] optimization problem and can't be solved for large networks. But, recent results [51, 121, 122, 129, 178] allow us to address this issue by rephrasing Eq. (7.3) to be a convex optimization problem as

$$\begin{aligned} \mathbf{L} &= \arg\min_{\mathbf{L}_0} \|L_0\|_*, \\ \text{s.t. } &\mathcal{P}_\Omega(\mathbf{D}) = \mathcal{P}_\Omega(\mathbf{L}_0), \end{aligned} \tag{6.6}$$

where $\|\mathbf{L}_0\|_*$ is the nuclear norm, or sum of the singular values of $\mathbf{L}_0$. This extends the solution to large networks that can be solved by splitting techniques and iterative matrix decomposition algorithms [121, 129, 178].

It is important to note that applying either Eq. (7.3) or Eq. (7.4) directly to a partially observed distance matrix *does not guarantee that the resulting low-rank matrix* ***L*** *is itself a distance matrix.* In particular, solving those equations does not guarantee that the resulting **L** will satisfy the triangle inequality [137]. For example, in our previous work [87], we addressed this issue by performing low-rank matrix completion to an appropriate Gram matrix after double-centering a distance ma-

[1] A problem is NP-hard, if an algorithm for solving it can be translated into one for solving any NP(nondeterministic polynomial)-problem.

trix **D**. However, in the current context, we perform matrix completion in the distance domain under the constraints of computed bounds. As we will demonstrate in the next Section, that the use of appropriate distance bounds allows for better graph reconstruction while simultaneously guaranteeing that the resulting matrix satisfies the triangle inequality.

6.3.1 Matrix completion with bounds

We can rearrange the optimization problem in Eq. (7.4) by imposing upper and lower bounds of the missing entries as,

$$\begin{aligned} \mathbf{L} &= \arg\min_{\mathbf{L}_0} \|\mathbf{L}_0\|_*, \\ &\text{s.t. } \epsilon_{\mathbf{L}} \leq \mathbf{L}_0 \leq \epsilon_U, \end{aligned} \tag{6.7}$$

where ϵ_L and ϵ_U are the element-wise lower and upper bounds of L_0. Here, $\epsilon_U = \epsilon_L = \mathbf{D}_{ij}$ when ij-th distance in **D** is observed.

Now we formulate the bounds ϵ_L and ϵ_U based on hop-distances in the network. Consider a graph with N nodes, with each node characterized by a vector of VCs corresponding to its hop-distances to M anchors ($M \ll N$). VCs of node n_i is given by

$$\mathbf{P}_{(i)} = [h_{n_i A_1}, h_{n_i A_2}, \ldots, h_{n_i A_M}], \tag{6.8}$$

where $h_{n_i A_j}$ is the shortest hop-distance between node n_i and anchor A_j. Thus,

$$\left\{\mathbf{P}_{N\times M} = [h_{n_i A_j}]_{ij}; i = 1, \ldots, N, j = 1, \ldots, M\right\} \tag{6.9}$$

is the matrix of VCs for all the nodes in the network.

The following proposition provides upper and lower bounds of an arbitrary hop-distance $h_{n_i n_j}$ between nodes n_i and n_j:

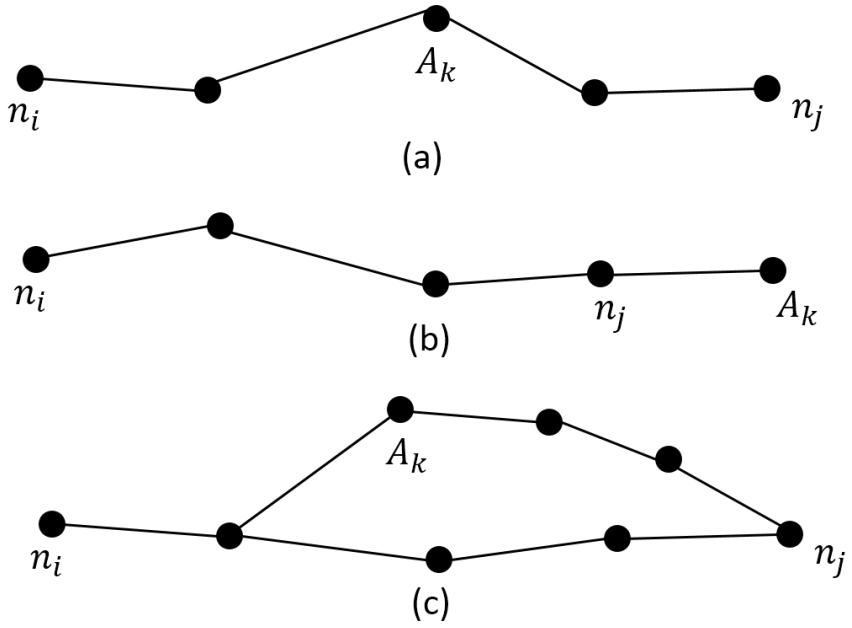


Figure 6.1: Possible network structures of two arbitrary nodes n_i, n_j, and an arbitrary anchor A_k. (a) Anchor lies on the shortest path between the two nodes and is located somewhere between the two nodes. (b) One of the node lies on the shortest path between the other node and the anchor. (c) The anchor is placed elsewhere.

Proposition 7. *For some i and j, the hop-distance $h_{n_i n_j}$ between two nodes n_i and n_j lies between following bounds:* $\max\left\{\max_{A_k\in\mathcal{A}}\left(\left|h_{n_iA_k}-h_{n_jA_k}\right|\right),1\right\}\leq h_{n_in_j}\leq\min_{A_k\in\mathcal{A}}\left(h_{n_iA_k}+h_{n_jA_k}\right)$.

Proof. We first generate the upper bound in the proposition. Any arbitrary anchor A_k can be located; a) on the shortest path of nodes n_i and n_j (the case that the anchor overlaps with a node is also considered here) [Fig. 6.1(a)]; b) collinear to n_i and n_j, and outside of the shortest path of nodes n_i and n_j [Fig. 6.1(b)]; or c) elsewhere than two above cases [Fig. 6.1(c)]. We develop the bounds upon these three cases. First, we assume that the anchor A_k is located on the shortest path of nodes n_i and n_j as shown in Fig.6.1(a). Then,

$$h_{n_in_j}=h_{n_iA_k}+h_{n_jA_k}. \tag{6.10}$$

If the anchor is located outside of the shortest path of n_i and n_j as shown in Figs. 6.1(b-c), the triangular inequality gives

$$h_{n_i n_j} < h_{n_i A_k} + h_{n_j A_k}. \tag{6.11}$$

By Eqs. (6.10-6.11), for all $n_i, n_j \in \mathcal{N}$ we have the upper bound

$$h_{n_i n_j} \leq \min_{A_k \in \mathcal{A}} \left(h_{n_i A_k} + h_{n_j A_k} \right). \tag{6.12}$$

Now we compute the lower bound in the proposition. If the anchor A_k is located on the shortest path of the nodes n_i and n_j [Fig. 6.1(a)], then

$$h_{n_i n_j} \geq |h_{n_i A_k} - h_{n_j A_k}|. \tag{6.13}$$

Here, only the equal sign remains, if the anchor overlaps with one of the nodes. If the anchor is collinear with nodes and outside of the shortest path of nodes [Fig. 6.1(b)], then,

$$h_{n_i n_j} = |h_{n_i A_k} - h_{n_j A_k}|. \tag{6.14}$$

Finally, we assume that the anchor is located, elsewhere than aforesaid two cases, as shown in Fig. 6.1(c). Then, the triangular inequality provides

$$h_{n_j A_k} < h_{n_i n_j} + h_{n_i A_k} \implies -h_{n_i n_j} < h_{n_i A_k} - h_{n_j A_k} \tag{6.15}$$

and

$$h_{n_i A_k} < h_{n_i n_j} + h_{n_j A_k} \implies h_{n_i A_k} - h_{n_j A_k} < h_{n_i n_j}. \tag{6.16}$$

By Eqs. (6.15) and (6.16),

$$h_{n_i n_j} \geq |h_{n_i A_k} - h_{n_j A_k}|. \tag{6.17}$$

By Eqs. (6.13), (6.14), and (6.17), for all $n_i, n_j \in \mathcal{N}$, we have

$$h_{n_i n_j} \geq \max_{A_k \in \mathcal{A}} \left| h_{n_i A_k} - h_{n_j A_k} \right| \tag{6.18}$$

Since n_i and n_j are two distinct nodes, $h_{n_i n_j} \geq 1$. Thus,

$$h_{n_i n_j} \geq \max \left\{ \max_{A_k \in \mathcal{A}} \left(|h_{n_i A_k} - h_{n_j A_k}| \right), 1 \right\}. \tag{6.19}$$

Eqs. (6.12) and (6.19) will prove the proposition. □

There are many standard libraries for solving optimizations problems like in Eq. (6.7). For example, using the library CVXPY [157] in the scripting language Python [158] the optimization problem in Eq. (7.4) can be solved using code such as described in Algorithm 6.

Algorithm 6 MC using Python toolbox CVXPY

```
import cvxpy
L = cvxpy.Variable(N, M)
objective = cvxpy.Minimize(cvxpy.norm(L, 'nuc'))
constraints = [ε_L ≤ cvxpy.L ≤ ε_U]
prob = cvxpy.Problem(objective, constraints)
result=prob.solve()
```

In Section 6.4, we will assess the performance of our algorithm for matrix reconstruction in both the following scenarios: (a) without the knowledge of bounds, and (b) with the knowledge of bounds, for missing entries.

6.4 Results for Undirected Social Networks

In this Section, we evaluate the effectiveness of the proposed technique in recovering the network characteristics. Three representative networks are used, a Facebook network with 744 nodes, a scientific collaboration network [Arxiv GR-QC (General Relativity and Quantum Cosmology)] with 4158 nodes, and an E-mail network from Enron with 4989 nodes. These network datasets are available in Stanford Network Analysis Project (SNAP) [179]. The characteristics of the three networks, namely size, diameter, clustering coefficient, and average length are listed in Table 6.5. The histograms of the hop-distances of these three networks presented in Fig. 6.2 show that the hop-distances in each network under experiment are normally distributed. They confirm that the VC generation can be done with a broadcast initiated from anchors with a low hop-count limit. Fig. 7.2 shows the logarithm of all the singular values of **D** for the three networks. As can be seen, there is a small set of singular values that are dominant indicating the low-rankness of **D** for these networks. This low-rankness provides us a platform to predict unknown entries of the distance matrix using MC techniques. All the code files for undirected networks can be found in Github at: code-link.

Table 6.2: Characteristics of three social networks: Facebook, Collaboration, and Enron Email.

Network	Size	Diameter	Clustering Coefficient	Average length
Facebook	744	7	0.637	2.5549
Collaboration	4158	17	0.556	6.0479
Enron Email	4989	8	0.484	3.139

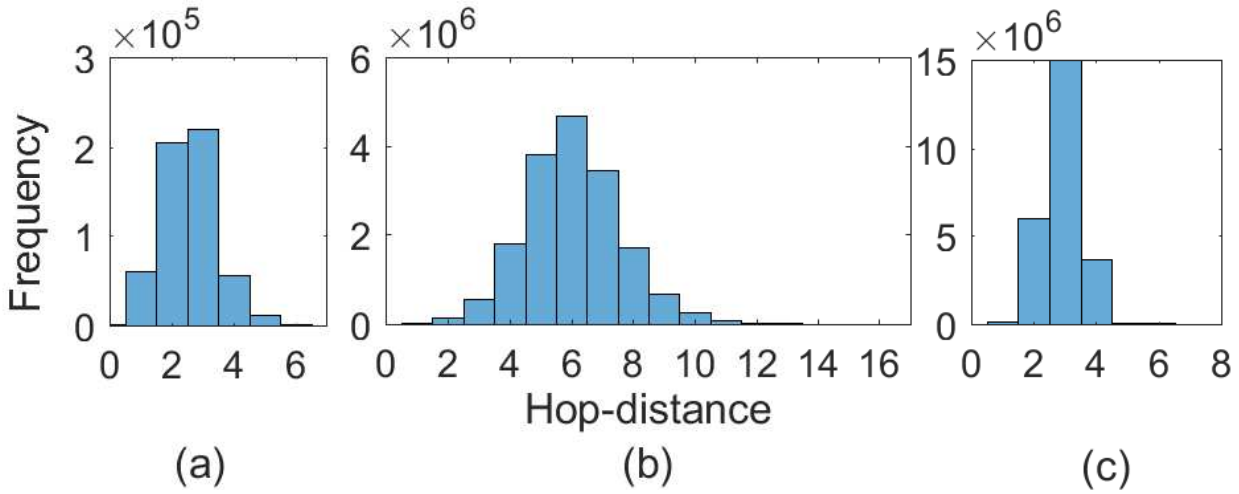


Figure 6.2: Histogram of the hop-distances for the social networks: (a) Facebook, (b) Collaboration, and (c) Enron E-mail.

6.4.1 Evaluation Methods

We evaluate our prediction method and bounds in two different ways. We use different sampling techniques to start from. This changes the input of the method. These two techniques can be described as below:

1. **Random VC-to-VC**: We select M columns from the complete $N \times N$ distance matrix. This $N \times M$ matrix is then randomly sampled for a certain percentage of its entries. Thus, our starting data looks like an $N \times M$ matrix with random entries missing. As the **D** matrix is symmetric in nature, for every ij^{th} that is missing we also eliminate the corresponding ji^{th} entry. We try to recover the original columns of $N \times M$ matrix by predicting the missing entries. This is illustrated in Fig. 6.4 [a].

2. **VC-to-full**: We select only M columns from the $N \times N$ matrix. We try to recover the complete $N \times N$ distance matrix from only these few columns. This is illustrated in Fig. 6.4 [a].

We will first look at the results for recovering a randomly sampled distance matrix by predicting the missing entries.

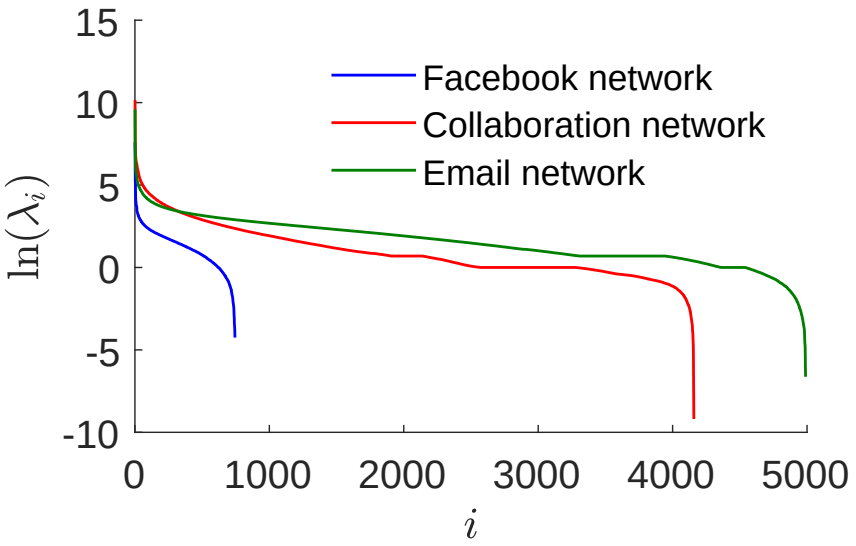


Figure 6.3: Natural logarithm of singular values $[\ln(\lambda_i)]$ of three distance matrices created over three social networks, namely, Facebook, Collaboration, and Email. These plots indicate that the distance matrices of given networks are naturally close to low-rank

6.4.2 Virtual Coordinate Matrix Formation

Virtual Coordinate matrix is a subset of complete distance matrix with the knowledge of node distances from only a few nodes, referred to as anchors. Low-rank MC is performed for three VC sets with a different number of anchors and a different number of missing entries. We evaluate the system performance with 50, 100, and 150 random anchors for the Facebook network; and 100, 150, and 200 random anchors each for Collaboration and Enron Email network. The number of anchors required to effectively capture the topological properties of the given network depends on the size, shape, and structure of the network [86]. Thus, we have tentatively chosen the anchor numbers for Facebook, Collaboration, and Email networks with their number of nodes in mind. Although a random selection of anchors does not guarantee a complete capture of all network characteristics, it provides a good approximation of the topology while using only a small percentage of the entire distance matrix. Now, let us consider the singular values of VC matrices and complete distance matrix for the three networks as shown in Fig. 6.5. Consider that, the predicted missing entries in **D** are rounded off to the closest integer. This shows that each of the VC matrices is low-rank. Thus, a representative VC matrix of a network can be used to identify the low-rankness

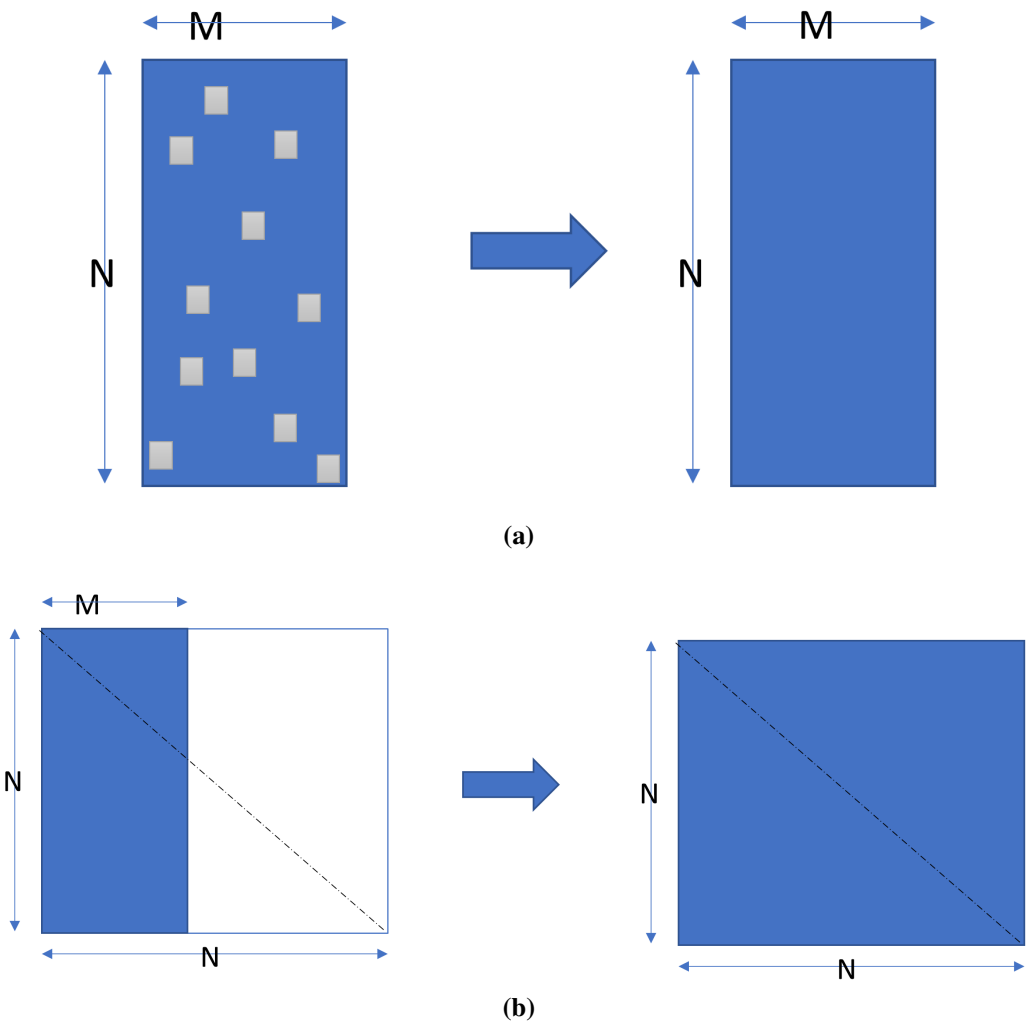


Figure 6.4: Evaluation methods: (a) Randomly sampling a Virtual Coordinate (VC) matrix to recover the original VC matrix. The gray area shows deleted entries and the blue area shows known entries. (b) Selecting only M columns from the complete $N \times N$ distance matrix to eventually recover the complete $N \times N$ distance matrix.

of **D** instead of analyzing the full distance matrix (Fig. 7.2). This is computationally more efficient than processing the entire **D**. Furthermore, the VCs necessary for the computation can be easily measured.

6.4.3 Performance Metrics

We use four performance metrics namely - mean error, absolute hop-distance error, average clustering coefficient, and average node degree to evaluate the accuracy of the proposed approach.

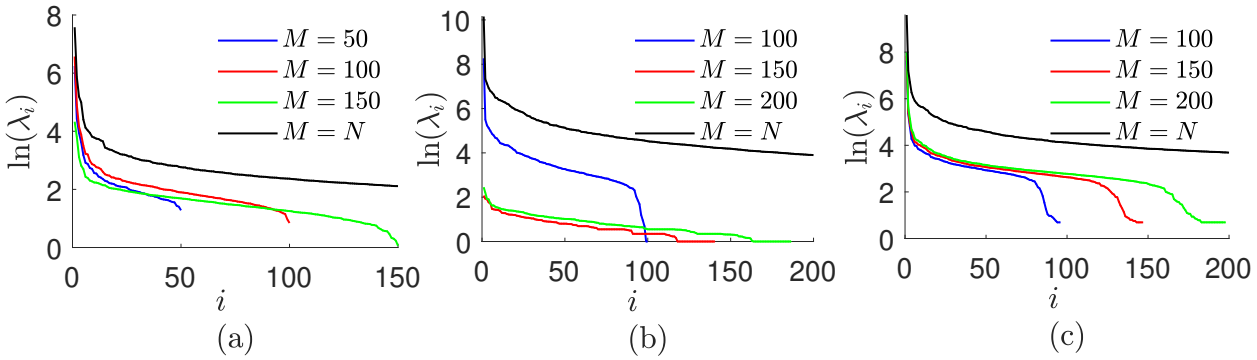


Figure 6.5: Natural logarithm of singular values $[\ln(\lambda_i)]$ for Virtual Coordinate matrices created from different number of Anchors (M) of three distance matrices created over three networks. Here (a) is for Facebook network, (b) is for Collaboration network, and (c) is for Email network. The steep slope at the beginning of the plot indicate low-rankness of the given Virtual Coordinate matrices.

These parameters are measured while regenerating missing entries from both cases, from complete distance matrix **D** and from VC matrix which is a subset of **D**. While mean and absolute hop-error gives a quantitative measure of the deviation of predicted values from actual, we use average clustering coefficient and the average node degree as the next two performance measures because they efficiently capture the intrinsic nature of the network. We measure these parameters before and after reconstruction to verify whether our reconstruction process preserves the very nature of the networks under test.

6.4.3.1 Mean Error

Mean error defined as

$$E = \frac{\sum_{i,j=1}^{N,M} |\mathbf{P}_{ij}(f) - \mathbf{P}_{ij}(0)|}{\sum_{i,j=1}^{N,M} \mathbf{P}_{ij}(0)} \times 100, \tag{6.20}$$

where $\mathbf{P}_{ij}(f)$ refers to the estimated VC matrix element of row i and column j when f fraction of random entries are missing from the given matrix. The mean error is a measure of percentage error in predicting the matrix entries.

We evaluate the effectiveness of estimating **D** from a random set of path lengths. This condition is simulated by randomly dropping a certain percentage of elements from complete **D**. We

randomly discard 10%, 20%, 40%, 60%, 80%, and 90% of entries from the original VC matrix. It simulates situations of missing data in social-networks or node failure in IoTs. These missing entries are estimated using the MC technique. Fig. 6.6 shows the mean error for aforesaid percentages of missing entries of the distance matrices for the listed three networks with and without bounds. We observe that the Facebook network has the highest error for all f values in both cases, while the Collaboration network has the lowest. Moreover, the mean error is lower irrespective of the network or f value when we work with bounds. This is convincing as the optimization scheme in Eq.(6.7) convergences better to the optimum as the bounds define a smaller feasible region than that with no bounds giving a more precise estimation from the data available.

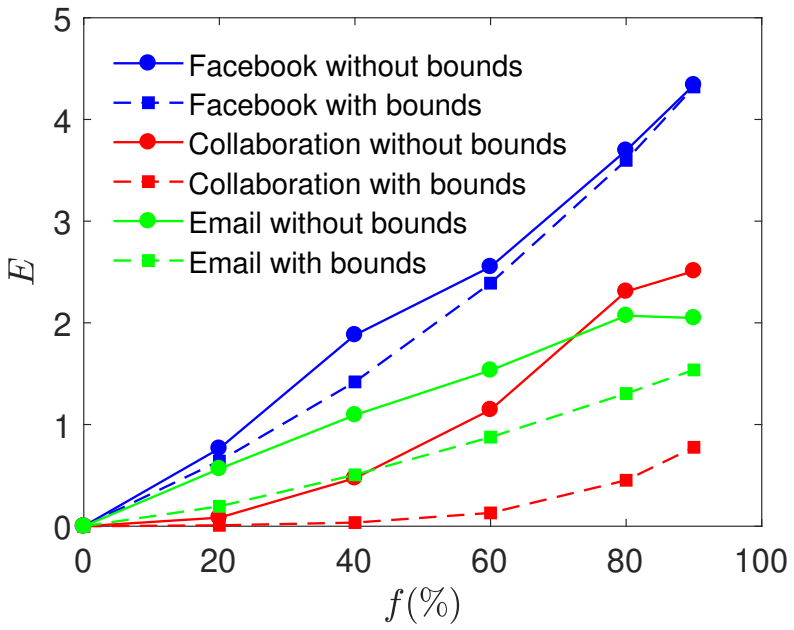


Figure 6.6: Mean error versus percentage of missing coordinates of distance matrix without bound (solid lines) and with bounds (dashed lines). Estimation performance is improved when the system is aware of upper and lower bounds for the missing entries.

Fig. 6.7 shows mean error for the networks under observation with respect to different numbers of anchors. We can see that the mean error increases when the percentage of missing entries increases. Note that mean error is relative to the sum of total hop-distances in a network. Thus, the cumulative sum of all the hop-distances in the network should be taken into consideration while

evaluating the quantitative value of mean error. This implies that the algorithm works to recover the missing entries of the network well with a quite low relative error in prediction even when only a subset of the distance matrix is provided.

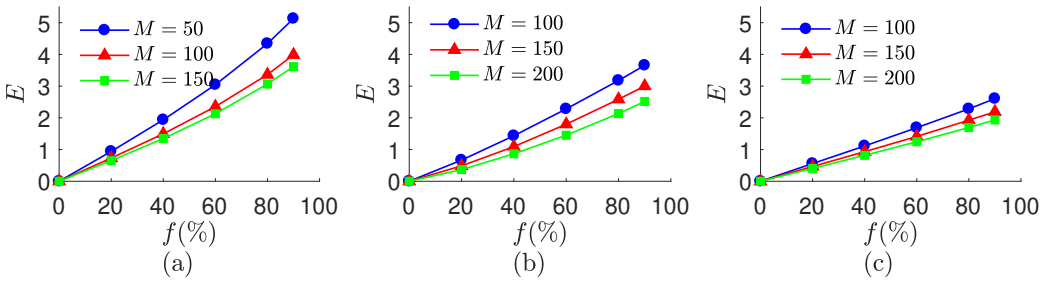


Figure 6.7: Mean error (E) versus percentage of missing coordinates (f) of VC matrix for different anchors. Here (a) is Facebook network, (b) is Collaboration network, and (c) is Email network. The more number of anchors, i.e., columns we start with, the more number of columns and thus number of node pair distances are captured and hence the prediction performance is improved with increase in number of anchors.

6.4.3.2 Absolute Hop-distance Error

The second metric is absolute hop-distance error, that captures average deviation in terms of hop-distances for the entire matrix. The absolute hop-distance error is defined as

$$H = \frac{\sum_{i,j=1}^{N,M} |\mathbf{P}_{ij}(f) - \mathbf{P}_{ij}(0)|}{MN}, \tag{6.21}$$

Product MN gives the total number of elements in the VC matrix.

Now we compute the absolute hop-distance error for the recovered matrix. Table 6.3 shows mean absolute hop-distance error versus percentages of missing entries for two cases with and without bounds. We observe that though this error measure increases with increasing f for both cases, the error for the bound case is smaller than that of the unbound case which bolsters our conclusion from section 5.2.1.

Table 6.3: Absolute hop-distance error versus percentage of missing coordinates of distance matrix without bound and with bounds

$f(\%)$	Facebook				Collaboration				Email			
	Without bounds		With bounds		Without bounds		With bounds		Without bounds		With bounds	
	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std
0	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00
20	.01	.10	.01	.07	.00	.02	.00	.02	.00	.06	.00	.04
40	.02	.14	.01	.11	.00	.03	.00	.03	.01	.09	.00	.06
60	.03	.17	.02	.14	.00	.05	.00	.04	.01	.11	.01	.08
80	.04	.20	.03	.18	.00	.06	.00	.05	.02	.13	.01	.10
90	.05	.21	.04	.19	.01	.08	.00	.06	.02	.14	.01	.12

Fig. 6.8 show absolute hop-distance error vs percentage of missing entries for the three networks. For the Facebook network of 744 nodes, from only 20% of the entries (i.e., 80% missing) and 50 (i.e., approximately 3%) random anchors, the VC set can be reconstructed with an absolute hop-distance error of four. The error is substantially lower for higher number of anchors. The performance is even better for the other two networks which are larger and more complex, thus showcasing the efficiency of the approach. These results show that the VC matrix can be reconstructed with high accuracy from only a fraction of the complete information.

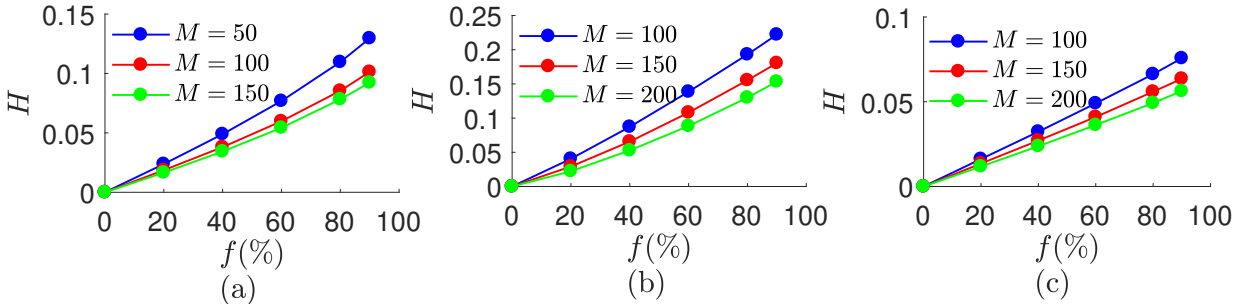


Figure 6.8: Absolute hop-distance error (H) versus percentage of missing coordinates (f) of VC matrix for different anchors. Here (a) is Facebook network, (b) is Collaboration network, and (c) is Email network. The more number of anchors, i.e., columns we start with, the more number of columns and thus number of node pair distances are captured and hence the prediction performance is improved with increase in number of anchors.

6.4.3.3 Average Clustering Coefficient

The clustering coefficient of a network is a measure of the degree to which nodes in a graph tend to cluster together. It is a measure of connected triples in the network. Two versions, global clustering coefficient, and local clustering coefficient, of this measure, exist. The global clustering coefficient was designed to give an overall indication of the overall clustering in the network, whereas the local parameter gives an indication of the embeddedness of single nodes. In this analysis, as we will be referring to the overall average, the global clustering coefficient of a network is used. This can be defined as

$$C = \frac{3 \times \text{number of triangles}}{\text{number of triples}}. \tag{6.22}$$

A triangle consists of 3 nodes that are completely connected to each other (i.e., a 3-clique) and a connected triple consists of three nodes $\{i, j, k\}$ such that node i is connected to node j and node j is connected to node k. The factor of 3 arises because each triangle gets counted 3 times in a connected triple. The clustering coefficient C indicates how many triplets are connected triangles. The minimum value of C is 0 with no vertex connected, while the maximum being 1 for every vertex being connected to every other vertex [180]. The observation is more interesting for the Email network where people exchange emails forming triads whereas it is relatively low in value for the Twitter network ($C = 0.19$) suggesting that friend of a friend phenomenon is not quite common when compared to that in Email network [181].

As we considered real-world networks for this study, they exhibit average clustering coefficients around 0.5. We can see in Fig. 6.9 that the average clustering coefficient for all the three networks varies within a 5% range of the original value for prediction without bound and within 10% for prediction with bound, thus, preserving the nature of the network after reconstruction.

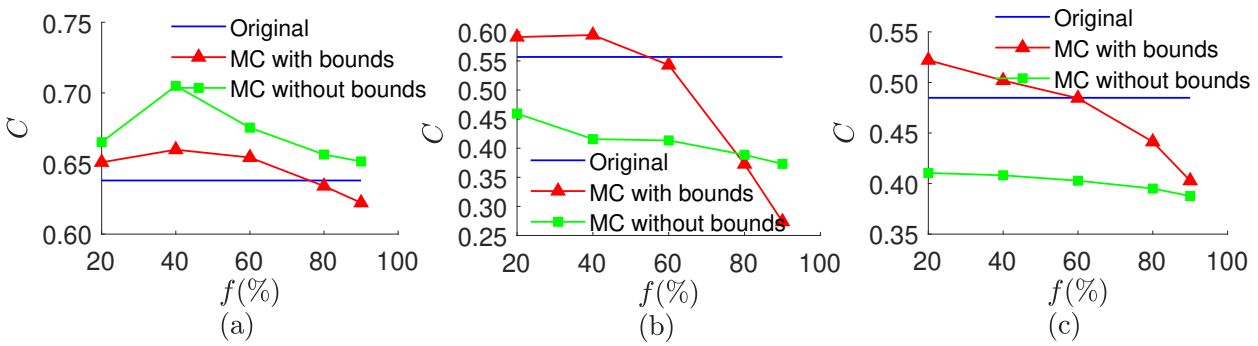


Figure 6.9: Average clustering coefficient (C) versus percentage of missing coordinates (f) for the original network, and recovered networks without bounds in MC (see green lines) and with bounds in MC (see red lines). Here (a) is for Facebook network, (b) is for Collaboration network, and (c) is for Email network. Clustering coefficient of the recovered matrix is closer to the original parameter value when upper and lower bounds for missing entries are known.

6.4.3.4 Average Node Degree

Topological characteristics in two situations of a given network can be compared by average node degree (ξ) and node degree distribution. The degree of a node is the number of edges connected to it. It is closely related to the density of a network. In a graph model, average node degree helps to quantify the probability of two nodes being connected [182]. The average node degree can be computed by

$$\xi = \frac{\sum_{i=1}^{N} \text{degree}(i)}{N}, \tag{6.23}$$

where N is number of nodes in the network.

Fig. 6.10 presents the effect of percentage deletion on average node degree for all three networks. We can see that the deviation in node degree is under 20 for Facebook network, under 5 for Collaboration network, and under 10 for Email network in worst case scenario of 90% deletion of entries. This shows that the algorithm retains the average node degree value for the collaboration network and preserves the intrinsic connection characteristics of the network after reconstruction well for the Email and Collaboration network whereas for the Facebook network, the algorithm is able to retain the node degree for average deletion percentages. We can see in Fig. 6.6 and Table 6.3 that knowledge of bounds allows a smaller deviation and reduced error value for both the

error metrics used. Though the reduction in error is small, we can see that the results are comparable. However, note that distance matrices completed with bounds satisfy triangle inequalities obtained from known entries to recover the missing data. Thus, bound constraints allow us to successfully perform matrix completion in the distance domain.

To evaluate the performance of the recovery technique on the complete network in comparison with each other. We observe the complete distance matrix for the network and delete a certain percentage (f) of entries. These missing entries are then calculated using MC in conjunction with bounds.

We discuss the recovery results for 60% deletion of entries in each network. The mean error is minimum for the Collaboration network and maximum for the Email network while varying between the magnitude of 2 to 12. As E is a measure of percentage deviation from the original values, the magnitude infers different errors for different networks. For instance, with the average hop-distance of 3.1 in the Collaboration network and an E value of 12% would mean deviation from 2.7 to 3.4 in prediction (range of 0.7). Whereas, with an average hop-distance of 6, an E of 5% would mean deviation from 5.7 to 6.3 (range of 0.6). Thus, we can see that the algorithm performs consistently for all the three networks under consideration.

The H error gives a more direct comparison of error with the absolute hop-distance error being less than 0.35 for each network.

The clustering coefficient can be seen to lower in value with increased missing entries. At 60% deletion, we can observe that Facebook, Collaboration, and Email perform better than the former network while conserving the original clustering coefficient value. It shows that the recovery technique performs better for larger networks.

The average node degrees of Facebook, Collaboration, and Email networks are less than 25% away from the original values.

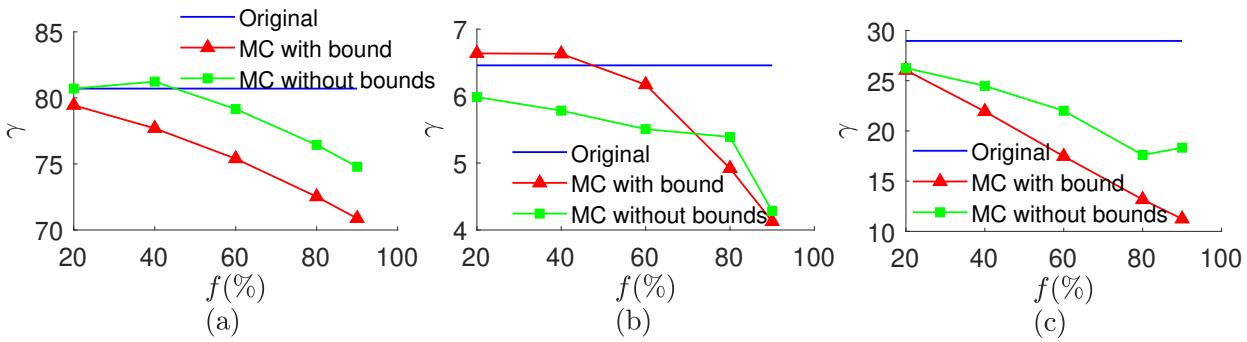


Figure 6.10: Average node degree (ξ) versus percentage of missing coordinates (f) for the original network, and recovered networks without bounds in MC (see green lines) and with bounds in MC (see red lines). Here (a) is for Facebook network, (b) is for Collaboration network, and (c) is for Email network. Average node degree of the recovered matrix is closer to the original network feature value when upper and lower bounds of the missing entries are known.

6.4.4 Recovering the complete Matrix from Virtual Coordinates

We saw in the previous Section how the Low-rank Matrix Completion technique coupled with derived bounds efficiently recovers a VC matrix from its random samples. We further test the prediction performance of this technique by trying to solve a harder problem. Here we select only M columns of the distance matrix **D** and try to recover the complete $N \times N$ matrix. To understand how this method performs with respect to the random sampling scheme, we find an equivalent random deletion percentage for given M columns and use these results for reference.

When we select M columns from a distance matrix of size $N \times N$, it is equivalent to deleting the P_r percentage of random entries from **D** matrix. P_r can be found by answering the question: *How much of the upper triangle, i.e., $(N \times N)/2$ entries, do we know when we select M columns?*. This can be visualized as shown in Fig. 6.11.

We are basically subtracting the green triangle from the $N \times M$ blue rectangle and calculating a ration of that with respect to the complete $(N \times N)/2$ triangle of the distance matrix. Hence, the known percentage of entries when M columns are selected can be presented as follows:

$$\textit{Known percentage} = \frac{[(M \times N) - (M \times M)/2] \times 100}{(N \times N)/2} \tag{6.24}$$

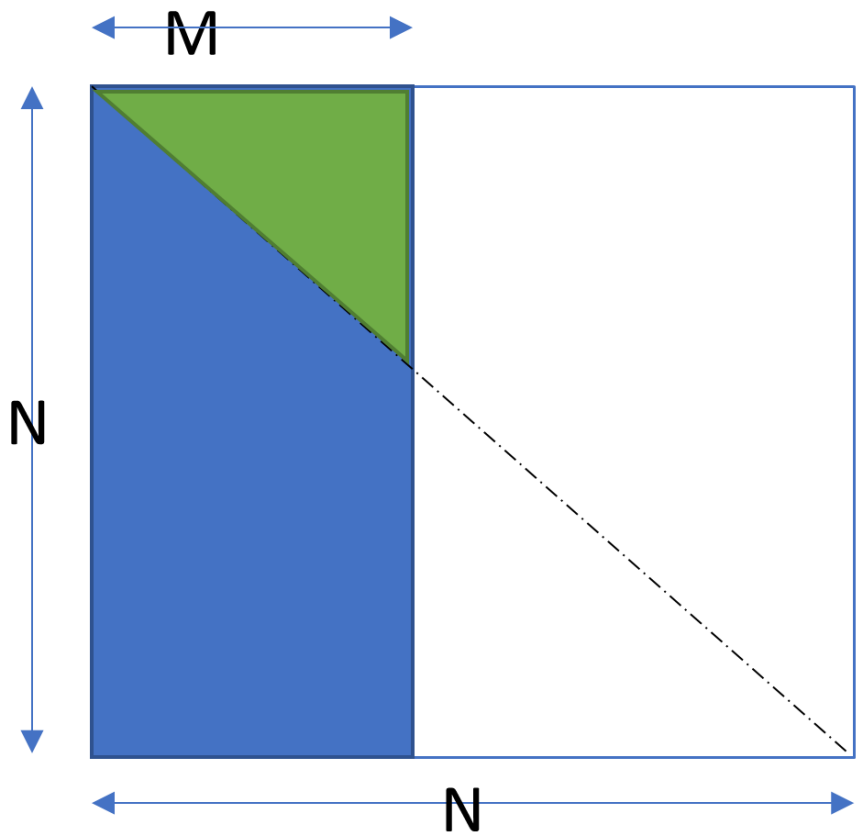


Figure 6.11: Illustration of percentage sampled when selecting M columns from an $N \times N$ matrix. The blue shaded portion are the new entries retained. Note that the green shaded portion has entries that are already present in the blue shaded region.

Eq. (6.24) can be further simplified into Eq. (6.25).

$$\textit{Known percentage} = \frac{[(M \times N \times 2) - (M \times M)] \times 100}{(N \times N)} \tag{6.25}$$

P_r can thus be represented as given in Eq. (6.26).

$$P_r = 100 - \textit{Known percentage} \tag{6.26}$$

Note, that as the given sample of M columns of **D** matrix contain approximately the same number of entries as when P_r percentage is deleted from matrix **D**, now the two evaluation schemes,

namely random VC-to-VC and VC-to-full, can be compared fairly. All the code files for undirected networks can be found in Github at: code-link.

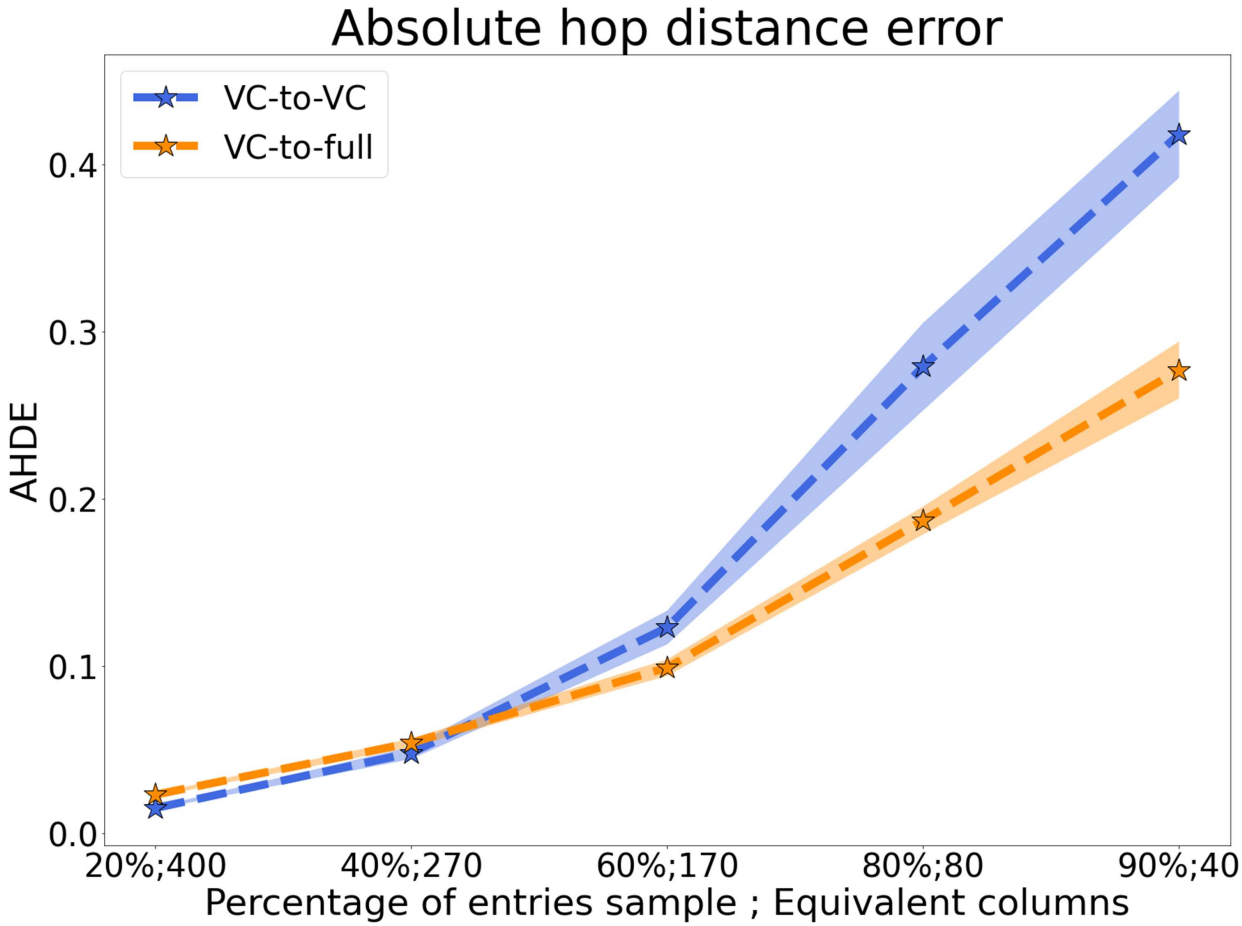


Figure 6.12: Absolute hop-distance error: Performance comparison: Recovery of randomly sampled VC matrix to complete VC matrix (VC-to-VC, see blue line) versus recovery of the complete **D** matrix from M columns (VC-to-full, see orange line). The X-axis gives the percentage of entries deleted and equivalent number of columns selected whereas the Y-axis gives the magnitude of absolute hop-distance error. The shaded region shows the standard deviation for each plot. VC-to-full stream is able to recover the original distance matrix with less absolute error as compared to the VC-to-VC stream.

Fig. 6.12 and Fig. 6.13 present results for random VC-to-VC and VC-to-full recovery methods. The X-axis gives the percentage of entries deleted and an equivalent number of columns selected whereas the Y-axis gives magnitude of the error. We can see that AHDE, as well as mean error performance of the LMC, integrated with bounds technique is slightly better, i.e., the error mag-

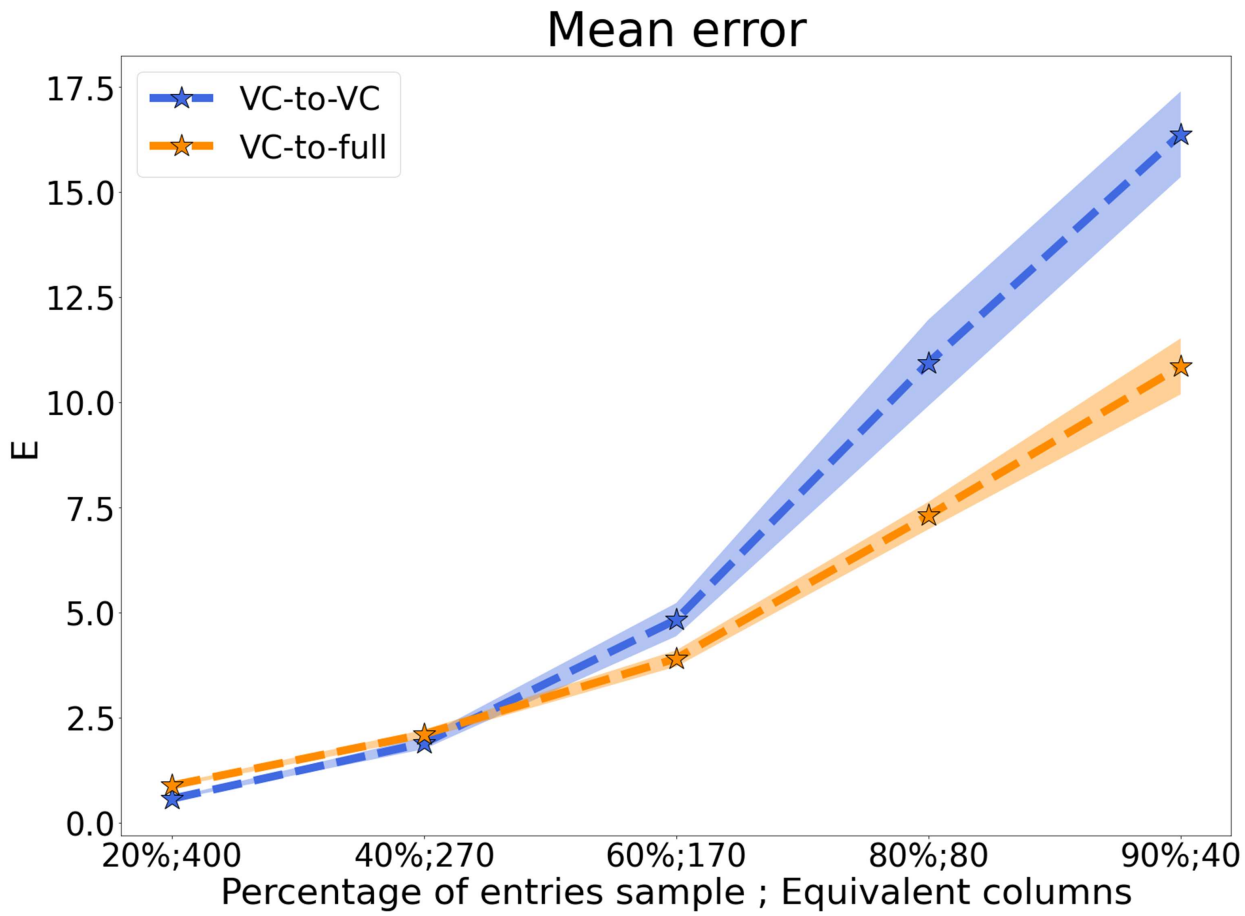


Figure 6.13: Mean error: Performance comparison: Recovery of randomly sampled VC matrix to complete VC matrix (VC-to-VC, see blue line) versus recovery of the complete **D** matrix from M columns (VC-to-full, see orange line). The X-axis gives the percentage of entries deleted and equivalent number of columns selected whereas the Y-axis gives the magnitude of mean error. The shaded region shows the standard deviation for each plot. VC-to-full stream is able to recover the original distance matrix with less mean error as compared to the VC-to-VC stream.

nitude is slightly smaller, for random VC-to-VC stream when less than 40% of entries are deleted or equivalent columns are selected to contain more than 60% of entries. However, as fewer entries are available, VC-to-full stream performs better. This is so because while selecting M columns from an $N \times N$ matrix, we start with M, completely sampled, columns. This gives us the distances from M nodes to all the other nodes in the network.

Fig. 6.14 and Fig. 6.15 present results for random VC-to-VC and VC-to-full recovery methods. Here we are computing the clustering coefficient and the average node degree of the recovered

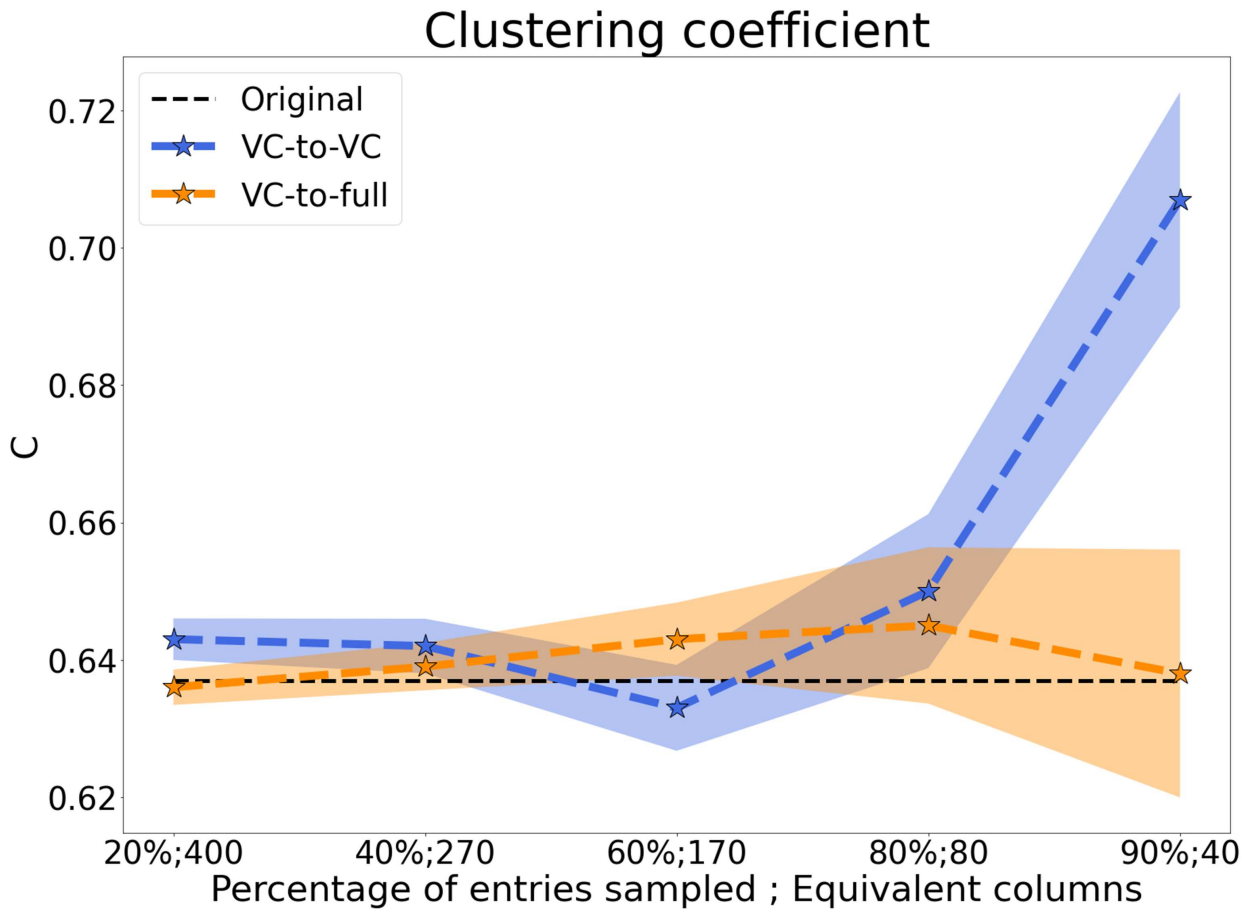


Figure 6.14: Clustering Coefficient: Performance comparison: Recovery of randomly sampled VC matrix to complete VC matrix (VC-to-VC, see blue line) versus recovery of the complete **D** matrix from M columns (VC-to-full, see orange line). The X-axis gives the percentage of entries deleted and equivalent number of columns selected whereas the Y-axis gives the magnitude of clustering coefficient of the recovered distance matrix. The shaded region shows the standard deviation for each plot. VC-to-full stream is able to recover the original distance matrix with more stability and less variation around the original clustering coefficient (see black dotted line) value as compared to the VC-to-VC stream.

network. The X-axis gives the percentage of entries deleted and equivalent number of columns selected whereas the Y-axis gives the magnitude of clustering coefficient (C) and average node degree (A) respectively. The original value of the clustering coefficient for this Facebook network is 0.63 and the average node degree is 80.7. We can see that both the parameters are more stable in the predicted network for VC-to-full stream. Whereas for the random-VC-to-VC stream, the recovered parameter value varies greatly with the number of observed entries. Though the values are stable, C value varies close to the original magnitude of 0.63 for VC-to-full stream and **A** value also stays close to its original magnitude of 80.7. Parameter values for random VC-to-VC stream

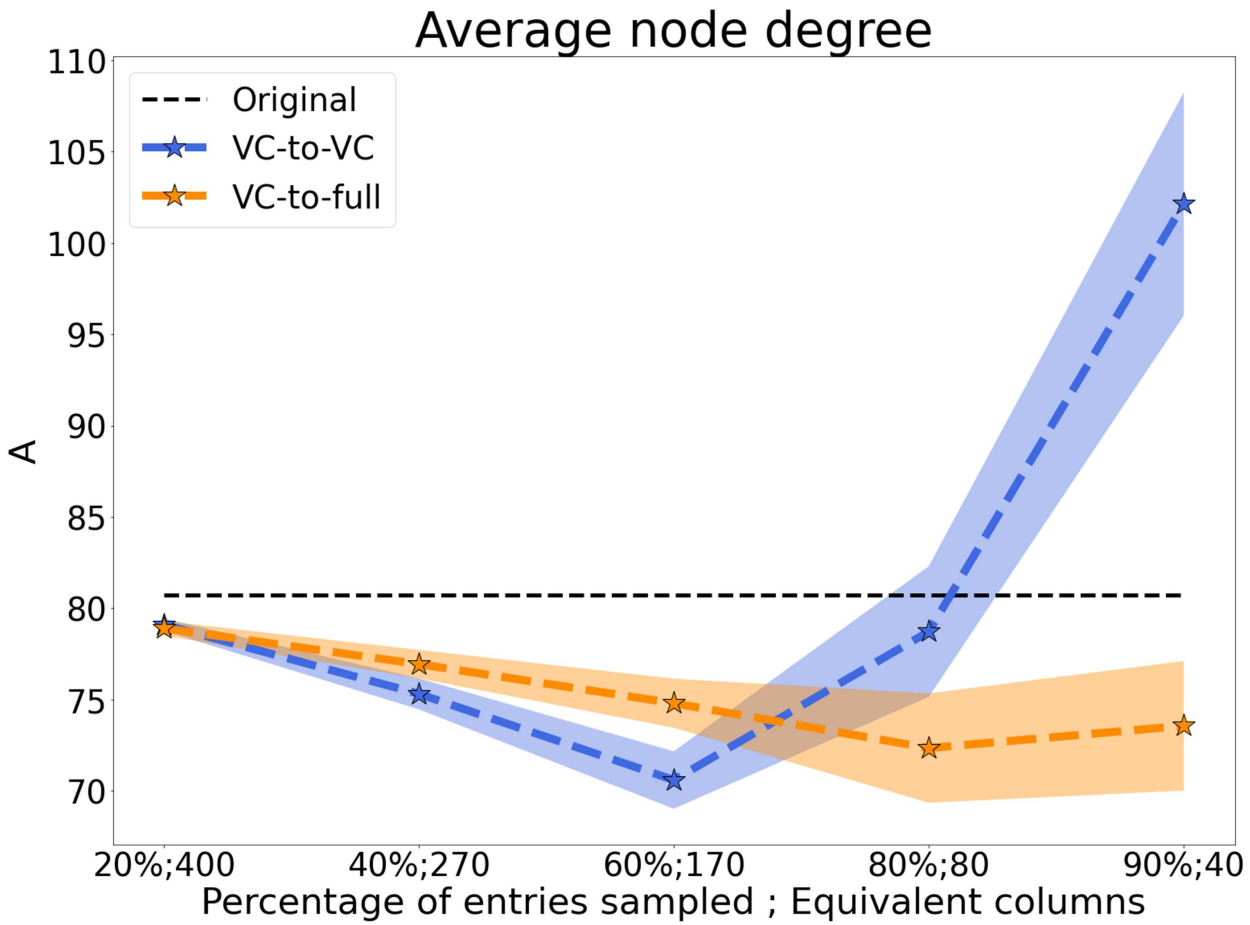


Figure 6.15: Average node degree: Performance comparison: Recovery of randomly sampled VC matrix to complete VC matrix (VC-to-VC, see blue line) versus recovery of the complete **D** matrix from M columns (VC-to-full, see orange line). The X-axis gives the percentage of entries deleted and equivalent number of columns selected whereas the Y-axis gives the magnitude of average node degree of the recovered distance matrix. The shaded region shows the standard deviation for each plot. VC-to-full stream is able to recover the original distance matrix with more stability and less variation around the original average node degree (see black dotted line) value as compared to the VC-to-VC stream.

are observed to show a positive trend with the increase in deletion percentage. This is so because when large number of random entries are missing, more and more missing entries are considered to have a lower bound of 1, i.e., the missing entries hold lose bounds due to missing common anchor of reference. On the contrary, when M columns are selected, we always have a common anchor node to compute the lower bound and it will always be ≥ 1 and thus a lower bound of higher than 1 is feasible for VC-to-full stream for all the missing entries. We cannot guarantee the same for random VC-to-VC.

We further compare our prediction performance by comparing the distributions of original and the recovered distance matrices. Kullback-Leibler divergence, D_{KL} [183] is a way to calculate distance between two distributions. It is also known as *"Relative entropy"*. It is applied to measure Shannon's relative entropy in information systems or information gain when comparing statistical models of inference. Given distributions P and Q over X and Q absolutely continuous with respect to P, the Kullback-Leibler divergence of Q from P is the P expectation of $(-\log_2\{P/Q\}$. Thus the Kullback-Leibler divergence, D_{KL} can be expressed as given in Eq. (6.27).

$$D_{KL}(P,Q) = -\int_X \log_2 (Q(x)/P(x))dP) \tag{6.27}$$

Fig. 6.16 shows the D_{KL} for distance distribution between original and recovered for both random VC-to-VC stream (blue line) and VC-to-full stream (orange line). We can see that the KL is stable for VC-to-VC stream. This implies that the distance distribution of recovered distance matrix is not much dependent of the number of entries sampled. On the contrary, KL for random VC-to-VC stream varies a lot with the number of sampled entries.

6.5 Summary for Undirected Network Prediction

A low complexity technique to capture social network topology has been presented. The algorithm uses a small set of path measurements, obtained by randomly deleting a percentage of entries or by measuring distances to a set of anchors, and constructs the distance matrix from this information. Matrix completion is applied to the incomplete matrix to recover the complete topology of the network. System performance has been evaluated based on the error in distance matrix prediction and how well the basic characteristics, such as - node degree and global clustering coefficient, of the networks are preserved even after recovery from partial information. Three networks have been selected for the experiment - Facebook, Collaboration, and Email network for their low-rank distance matrix, small-world nature, and naturally evolved structure from human communications.

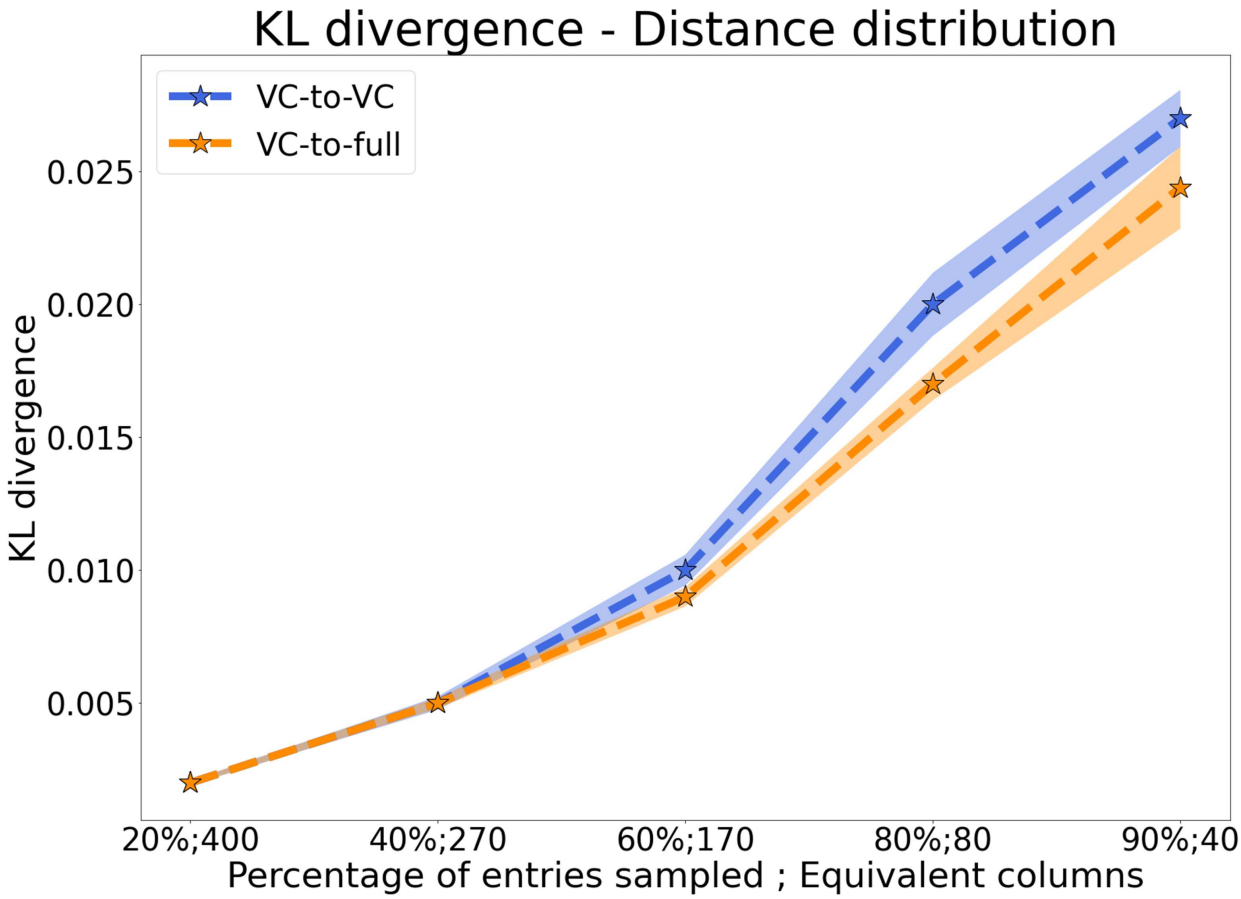


Figure 6.16: KL divergence for distance distribution: Performance comparison: Recovery of randomly sampled VC matrix to complete VC matrix (VC-to-VC, see blue line) versus recovery of the complete **D** matrix from M columns (VC-to-full, see orange line). The X-axis gives the percentage of entries deleted and equivalent number of columns selected whereas the Y-axis gives the magnitude of KL divergence between the distributions of original and recovered distance matrices. The shaded region shows the standard deviation for each plot. VC-to-full stream is able to recover the original distance matrix with less KL divergence as compared to the VC-to-VC stream.

The listed results show that the network reconstruction algorithm performs with a mean error of well within 5 percent and an absolute hop-distance error of less than one hop. It also maintains the global clustering coefficient of the reconstructed network under 10-15 percent variation while the node degree is preserved with better accuracy. This indicates that the listed technique predicts with high accuracy.

Upper and lower bounds for the missing entries in the given distance matrix have also been derived. The results show that bounds allow a better prediction with lower mean and absolute hop-distance error while satisfying triangle inequality during matrix completion. Though the deviation

in clustering coefficient and average node degree is greater than that without bounds, the nature of the network is preserved during MC in the distance domain. It was interesting to see how recovering the complete distance matrix from a few of its columns can be more effective than random sampling, even when only a small number, M, of columns are sampled. This further illustrates the importance of knowing tentative upper and lower bounds for missing entries during Low-rank Matrix Completion.

Evaluation using these three representative networks shows that the proposed matrix completion based technique can capture the topological properties of complex networks accurately. The results also lead to the conclusion that the distance from each of the nodes to a set of random anchor nodes can serve as a compressed representation of a network. Such a representation can be obtained relatively easily compared to capturing all the connectivity information, and with the proposed technique, it can be used to construct the complete network topology fairly accurately.

Even though we have shown only three networks in this article, the algorithm can be applied to any network with a low-rank distance matrix. Thus, the presented algorithm offers many applications. It can be used for storing network data in compressed format at each node in distributed networks and can be reconstructed as required. The algorithm can also be used for predicting missing entries in any graph-based models such as - recommender systems, image restoration, protein topology prediction, etc. While considering a dynamic network that evolves with time, the presented technique can be deployed for link prediction as well. Also, being able to predict bounds for a missing entry can help make better and more meaningful estimations even with approximation.

Extracting connectivity information in massive social networks is important for many applications. Algorithms developed for undirected networks cannot be used with social networks characterized by directed edges. We present a method to extract the network topology from a small sample of distance measures without the need for exhaustive measurements. Tolerating missing data is also necessary when certain nodes participate in message passing, but are not accessible for direct measurements. An anchor-based sampling approach is proposed and compared with random sampling. As we demonstrate, real-world directed social networks such as Twitter and US election

blogs have hop-distance matrices that are low-rank. Low-rank matrix completion techniques are thus used to recover the complete topology from a relatively small set of measurements. Evaluation of the proposed technique using metrics such as distance distribution, degree distribution, and hop-distances show that the proposed technique is effective even when only a small fraction of distance entries are available.

6.6 Low-rank Matrix Completion for Directed Social Networks

Mining and extracting information from social networks are increasingly important challenges. Many of these networks have been rapidly increasing in size, e.g., Facebook had around 1.59 billion active users in 2016 [15]. Efficient and scalable methods are thus necessary for measuring and capturing information in such networks with manageable complexity in time, computation, and storage. Many large-scale real-world social networks have asymmetric relationships that can only be represented by directed links. Directed graph analysis is used, e.g., in ranking webpages among a sea of hyperlinks, detecting communities in Twitter network [184], understanding transactions in financial digraphs for better investments [185], processing phone call networks to get insights on traffic, investigative graph search for homeland security and intelligence [186], and inferring disease spread patterns in communities to deploy preventive measures [187]. Efficient techniques for directed networks have also gained importance in a wide array of fields such as biology, social psychology, and, of course, computer science.

Analysis of directed graphs, in general, is more complex than that of undirected graphs. Adapting techniques developed for undirected graphs [87] pose significant challenges as the nature of relationships captured by the directed edges are fundamentally different from those in undirected settings [24]. Undirected graph methods fail to capture the asymmetric relationships implied by the edges of a directed network captured by an adjacency matrix, the Laplacian, etc. [188] [189].

As storing and processing complete network connectivity measurements can be computationally expensive, statistical sampling techniques [190] are sometimes used in networks to extract a small subnetwork for experiments. However, due to restricted access (e.g., private modes, en-

crypted communication, or inaccessibility) and measurement limitations such subnetworks leave us with incomplete network measurements and inaccurate inferences.

Partially observed data also causes problems when analyzing large directed networks [191]. Additionally, missing data may alter the representative network significantly to affect network or node feature values thereby creating a gap between network data and meaningful network analysis. Thus, despite the growing need and applications of network analysis, and evolution in network analysis techniques, large network size, and incomplete data sets still inhibit effective inference and decision making based on directed networks.

Accurate estimation techniques to complete available partial data in accordance with the original network characteristics would bridge this gap leading to improvements in many network analysis schemes without burdening the measurement and communication infrastructure. Being able to sample a network to accurately recover it will also allow inexpensive storage of large networks and assist in communication of such information in distributed systems where resources are limited.

6.6.1 Contribution

When the objective is to use a relatively small sample network S with properties similar to that of the original network G, it is referred to as “scale-down" sampling. This technique, however, changes the original feature values for the given network [20]. Accordingly, we present here, a novel way to efficiently process and store large directed networks using a partial set of hop-distances. Thus, our approach does not use a small sampled network representing the original network, but a set of sample distances of the entire network. We present a novel network topology sampling scheme followed by an approach to recover the missing network connections. Our technique is based on Low-rank matrix completion (LMC), and thus we also demonstrate that many of the real-world social networks have low-ranked distance matrices. We employ the directed network communication infrastructure for data collection, yet if and when necessary the unmeasured facts can be accurately estimated from the sampled data.

Efficiency of the recovery technique is demonstrated on two real-world directed social networks, namely Twitter and an US election blog webpage network. We also evaluate the method with a network of Linux system files to demonstrate the applicability of the presented technique beyond social networks.

6.7 Related Work

This section reviews network topology capture and recovery techniques. There is a rich literature that suggests ways to capture information from social networks with web crawlers [174, 176]. Some of the common graph sampling techniques on social networks employ either the graph traversal method or the random walk method. Questionnaires, surveys [11], crawling [12], random sampling, snowball sampling, and salting [13] [14] are some of the other widely used data collection techniques. Owing to massive size of such graphs, significant computing, and memory resources are needed to capture data. Addressing the effects of missing data on network features, [191] proposes a Bayesian data augmentation scheme for analysis focused on Exponential Random Graph Model (ERGM) when an influential attribute is partially observed with certain assumptions. Recovery of edge entries in directed graphs is presented in [192] using robust rank-k matrix completion where k is a user-defined parameter, based on their work using adjacency matrix. Recovery of the underlying Euclidean metric given a directed graph and the underlying dimension *d*, is presented in [51]. Note that, not all networks can be interpreted in terms of Euclidean distances. In networks such as Facebook, webpages, and hyperlinks, where physical distances do not make much sense, but hop-distances are of paramount importance in understanding the network topology. Recovery of one or multiple graph signals using the alternating direction method of multipliers is proposed in [193]. A recovery algorithm for chain graphs is presented in [194]. A technique for predicting links by blockmodeling graphs is presented in [195]. There has also been work that leverages low-rank properties of a network to detect intrinsic network structures in [196]. However, these techniques are not directly applicable to directed networks.

The problem of using partial hop-distances to recover the unknown entries in the network had not been addressed yet for directed graphs.

6.8 Theory

In this Section, we present the notation and terminology, demonstrate the low-rank nature of the distance matrix of real-world networks with directed edges, and propose network sampling and recovery schemes.

6.8.1 Network representation

We focus our attention on *strongly connected and unweighted directed graphs* G defined by $G = \{V, E\}$, where V is the set of nodes or vertices and E is a set of directed edges, corresponding to communication links, friendship status on a social network. Such a graph may be represented by an *adjacency matrix* $\mathbf{A}$ [143], where

$$\mathbf{A} = [a_{ij} = 1 \text{ if } (i,j) \in E,\ 0 \text{ otherwise }]. \tag{6.28}$$

Note that, $a_{ij} \neq a_{ji}$ when G is directed. Our main object of study will be *Hop-Distance* Matrices (HDMs), $\mathbf{H} \in \mathbb{N}_0^{N \times N}$, (where $\mathbb{N}_0$ denotes the non-negative integers $\mathbb{N} \cup \{0\}$), and

$$h_{ij} = \text{length of the shortest path from } n_i \text{ to } n_j. \tag{6.29}$$

$\mathbf{H}$, by construction, is asymmetric for directed graphs whereas it is symmetric for undirected graphs. $\mathbf{H}$ is invariant to the situation where two nodes have multiple paths between them of the same shortest length. The notations used are listed in Table 7.1.

It is known, that $\mathbf{H}$ is low-rank for many classes of undirected networks [85, 87], but it is not known whether this is the case for directional networks. We present results in Section 7.4 that demonstrate this to be the case for many large directed networks of interest as well. Low-rankness

Table 6.4. Notation

Notation	Description
$\mathcal{N}$	Set of nodes
N	Number of nodes, $\lvert\mathcal{N}\rvert = N$
$n_i \in \mathcal{N}$	i-th node (current node)
$\mathcal{A} \subset \mathcal{N}$	Set of anchor nodes
M	Number of anchors, $\lvert\mathcal{A}\rvert = M$
$A_i \in \mathcal{A}, i = 1, \ldots, M$	i-th anchor
h_{ij}	Minimum hop-distance between nodes n_i and n_j
$\mathbf{H}$	Hop-distance matrix
$\mathbf{A}$	Adjacency matrix
$\mathbf{P}$	Sampled hop-distance matrix
$\hat{\mathbf{P}}$	Predicted hop-distance matrix
λ_i	i-th singular value

of a distance matrix implies a significant dependence of distance vectors, i.e., rows of **H**, on each other. Consequently, low-rank matrix completion techniques may be used, e.g., to predict random missing distance entries. We exploit the low-rankness of **H** to provide systematic and practical approaches for sampling networks and recovering the full connectivity and topology from those samples.

6.8.2 Sampling schemes for directed social networks

Our goal is to capture the topology accurately using efficient network sampling schemes. A sampling scheme results in a subset of entries of **H**, which we represent by entries in a $N \times N$ matrix **P**. Thus, if the distance from i to j is measured by the sampling scheme, ijth element of **P** is equal to h_{ij}. The remaining elements of **P** are unknown or missing and thus, are denoted by $\mathcal{X}$. A recovery method is needed to obtain **H** from **P**, which consists of a small number of easily measurable distances.

Prominent among hop-distance based sampling methods used for undirected networks are those relying on anchor-based Virtual Coordinate Systems (VCS), in which each node is represented by a Virtual Coordinate (VC) vector corresponding to the *minimum* hop-distances to a small set of

nodes, known as *anchors* [117–119]. We follow the notation in [85] and consider networks where M of the N nodes are designated as "anchors". Without loss of generality, we assume that nodes $1..M$ are anchors and denote them by $A_k, k = [1, 2, .., M]$. An anchor-based method is especially relevant for social networks where messages generated by a small set of prominent nodes are forwarded to all or most of the network nodes. One possibility is to select such prominent nodes as anchors and keep track of the number of hops as their messages propagate to other nodes. Selection of a random set of nodes as anchors is another less complex solution. Thus, the hop-distance from each of the M anchors to any given node is known, i.e., all the elements of the M rows of **P** corresponding to the set of anchors nodes are known. We also may record paths from these M nodes to each of the N nodes giving us additional intermittent entries as illustrated in Fig. 8.2. Such measurements are referred to as *Intermediate Path Lengths* (IPL) and the nodes on the path are *Intermediate Path Nodes* (IPN). Therefore, **P** may be written as

$$\mathbf{P} = \begin{bmatrix} h_{A_1:} \\ \vdots \\ h_{A_M:} \\ h_{M+1:} \\ \vdots \\ \vdots \\ h_{N:} \end{bmatrix}_{N \times N} \tag{6.30}$$

where $h_{A_i:}$denotes the i-th row for $i = [1, 2...M]$ and $h_{i:}$ denotes the i-th row for $i = [M + 1...N]$. Thus, for anchor based sampling, $h_{A_k:} = [h_{A_k1}, h_{A_k2}, .., h_{A_kN}]$, $k = [1, 2, .., M]$. As mentioned above, many of the entries of rows $M + 1.., N$ remain unknown, and therefore are of the form $h_i := [h_{i1}, h_{i2}, \mathcal{X}, .., \mathcal{X}, .., h_{iN}]$. A sampling scheme needs to ensure that each row has at least one non-zero distance value, obtainable as IPLs in anchor based sampling. For large networks it is generally desirable to have only a small subset of nodes as anchors, i.e., $M \ll N$. The i-th row

is called the Virtual Coordninate (VC) vector of the i-th node, and j-th column corresponds to the hop-distance to the j-th nodes from other nodes in the network.

Algorithm 7 Generation of VC from anchors

procedure ANCHOR SELECTION
M random nodes self-select or are selected as anchors
end procedure

procedure AS AN ANCHOR NODE
Broadcast message to all its neighbors. It includes:
a) node number A_k,
b) hop count $h_{A_k} = 0$,
c) (optional) $TTL = t$ to limit propagation distance
d) fields for recording all or a subset of of nodes visited in the path (IPL).
end procedure

procedure AS AN INDIVIDUAL NODE $i[i = 1..N]$
Note: P_{ki} is a variable that stores the shortest hop-distance from A_k to n_i, initialized to large value;
For each Message (A_k, h_{A_k}, TTL, IPN) received
if $P_{ki} > h_{A_k}$ **then**
$P_{ki} = h_{A_k}$; $h_{A_k} = h_{A_k} + 1$;
$TTL = TTL - 1$;
if $TTL > 0$
Insert i in IPN
Forward message to neighbors
end if
end if
end procedure

Algorithm 7 describes steps for generating VCs using messages originating from different anchor nodes. This approach may be implemented on a network, e.g., Facebook by broadcasting a message to friends and in turn encouraging them to forward the message. A Time-To-Live (TTL) t may be set to limit the messages in the network, which may result in unknown entries on rows in **P** corresponding to anchor nodes. This is acceptable for the recovery scheme provided that there is at least one known distance in each row and each column. Social networks classically abide by the "six degrees of separation" [177], implying that any two people are six or fewer connections away

from each other. Thus, setting the hop count to an integer equal to six would capture much of the information needed.

We compare this sampling scheme with random sampling which is widely used for scale-down sampling. These sampling schemes are described as follows:

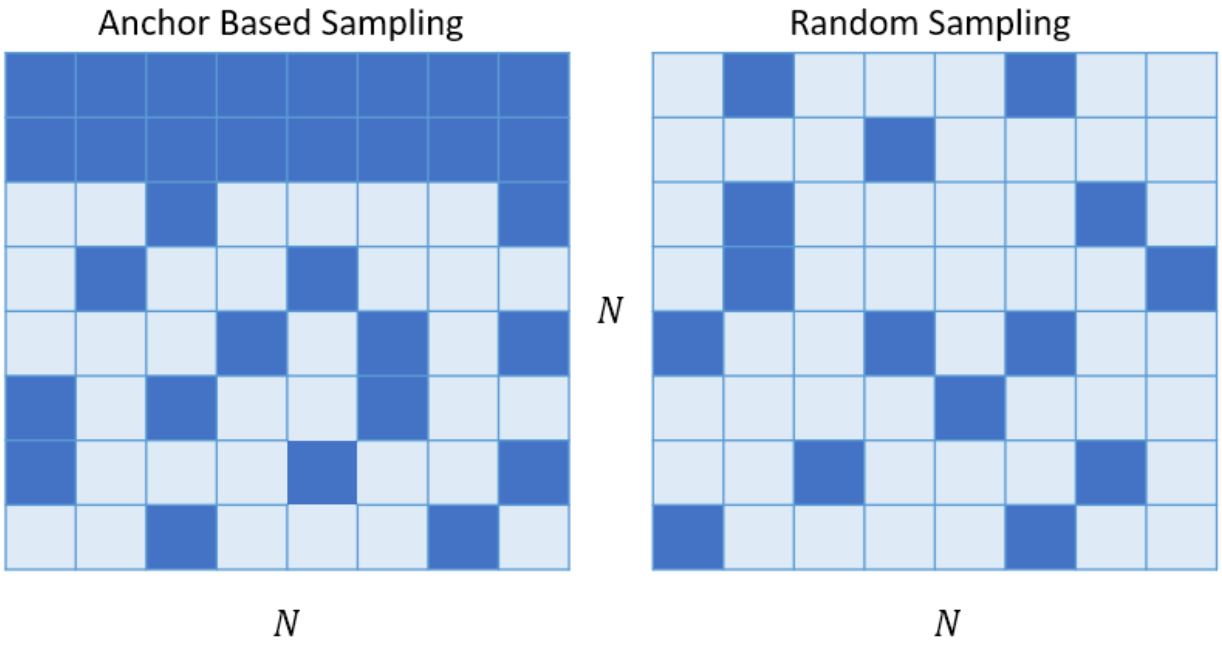


Figure 6.17: Visual illustrations of **P** for anchor based and random sampling schemes for $N = 8$. Known entries are denoted by dark blue and unknown values by light blue.

1. *Anchor based sampling:* A percentage of nodes are selected as anchors and paths from these anchors to other nodes are observed along with the IPLs.

2. *Random sampling:* A percentage of hop-distances are measured within the network among random node pairs.

These sampling schemes are illustrated in Fig. 8.2. Random walk based sampling works well for scale-down approaches [197]. However, as random walks are biased towards nodes with higher degrees, the produced samples are not representative of the original network when the network has non-uniform degree distribution. This causes a significant practical challenge since typical large-

scale, real-world, unstructured networks tend to have non-uniform degree distributions, e.g. power law degree distribution in unstructured peer-to-peer networks [198]. Thus, we select anchors and measurements randomly for an unbiased distributed sampling of network information.

6.8.3 Method: Network recovery technique

With the proposed network sampling techniques, some entries in $\mathbf{P}$ are *not observed* and the matrix $\mathbf{P}$ is therefore incomplete. Based on the low-rank assumption, we have leveraged modern ideas in low-rank *matrix completion* [51,123]. The key idea of such methods can be phrased as the following optimization problem:

$$\hat{\mathbf{P}} = \arg\min_{\mathbf{P}_0} \rho(\mathbf{P}_0), \;\; \text{s.t. } \mathcal{P}_\Omega(\mathbf{P}) = \mathcal{P}_\Omega(\mathbf{P}_0), \tag{6.31}$$

where $\mathbf{P}$ is a partially observed matrix, $\hat{\mathbf{P}}$ is the completed $\mathbf{P}$ matrix with all the missing entries predicted, and ρ is the rank operator. Given that Ω denotes a set of observed entries of the matrix $\mathbf{P}$ and $\mathcal{P}_\Omega$ denotes a projection of $\mathbf{P}$ onto Ω, the constraint imposes that the matrix $\mathbf{P}_0$ has the same values as $\mathbf{P}$ at the observed entries of $\mathbf{P}_0$. In other words, we seek to find a matrix $\mathbf{P}_0$ such that the rank of $\mathbf{P}_0$ [denoted $\rho(\mathbf{P}_0)$] is minimized while enforcing the constraint that the matrix we construct matches the input matrix at observed entries. Since we enforce the constraint $\mathcal{P}_\Omega(\mathbf{P}) = \mathcal{P}_\Omega(\mathbf{P}_0)$ the returned matrix $\mathbf{P}_0$ will be faithful to our measured hop-distances; but, $\mathbf{P}_0$ is free to take any values outside of Ω to minimize its rank. Unfortunately, as stated, Eq. (7.3) is an NP-hard optimization problem and that can only be solved for small networks. Recent research, [51, 123], under mild assumptions, allow the NP-hard optimization problem in Eq. (7.3) to be remodeled as a convex optimization problem

$$\hat{\mathbf{P}} = \arg\min_{\mathbf{P}_0} \|\mathbf{P}_0\|_*, \;\; \text{s.t. } \mathcal{P}_\Omega(\mathbf{P}) = \mathcal{P}_\Omega(\mathbf{P}_0), \tag{6.32}$$

where $\|\mathbf{P}_0\|_*$ is the sum of the singular values of $\mathbf{P}_0$, often called the *nuclear-norm* of $\mathbf{P}_0$. The optimization problem in Eq. (7.4) is *convex* and can easily be solved for millions of nodes on commodity computing hardware using splitting techniques and iterative matrix decomposition algorithms [121, 123]. Even though a matrix of distances such as $\mathbf{P}$ should not have negative entries, reconstructed matrix, $\hat{\mathbf{P}}$ does. As LMC is not fabricated to do that, we round-up all the negative values of $\hat{\mathbf{P}}$ to zero and positive values to the closest positive integer. There are many standard libraries for solving optimizations problems like in Eq. (7.4). Using the library CVXPY [157] in the scripting language Python [158] the optimization problem in Eq. (7.4) can be solved using code such as described in Algorithm 8.

Algorithm 8 Compute low-rank representation $\hat{\mathbf{P}}$ of the partially observed HDM $\mathbf{P}$.

Require: Partially complete hop-distance matrix $\mathbf{P}$.

1: **procedure** MATRIX COMPLETION
2: Compute recovered matrix $\hat{\mathbf{P}}$ using 7.4
3: Round-up all negative values of $\hat{\mathbf{P}}$ to zero and positive values to the closest positive integer
4: **end procedure**

There have been several research solutions on recovering missing entries of Euclidean distance matrices in directed graphs [51], however, to the best of our knowledge, recovering network data using partial hop-distances is a novel technique and has not been done for directed graphs such as social networks.

6.9 Results for Directed Social Networks

In this Section, we evaluate the effectiveness of the proposed technique in recovering the topology of directed networks. Three representative networks are used as listed in Table 6.5 [199]. These networks vary in size, characteristics, and are formed by different interactions. The Twitter network contains information about who follows whom on Twitter. Nodes represent users and an edge shows that the left user follows the right one e.g. the edge (i, j) indicates that user i follows user j. US election blogs network represents hyperlinks between blogs (nodes) in the context of

the 2004 US election. The Linux network is a network of Linux source code files, with directed edges denoting that they include each other. Thus, the collection represents social communication traits, information linking patterns of web pages, and software dependencies. Variety in nature of interactions in these networks is deliberately taken into consideration to evaluate the effectiveness of the presented topology retrieval technique on a broad range of applications.

Table 6.5. Characteristics of Twitter, US election blogs, and Linux Files networks

Network	Size	Diameter	Average node degree	Average shortest path
Twitter	157	9	10.13	3.29
US election blogs	793	8	39.8	3.19
Linux	913	18	9.12	5.91

6.9.1 Low-rankness in social networks

Here, we demonstrate the low-rankness of the three test networks. Fig. 7.2 shows the logarithm of all the singular values of **H**. As can be seen, there is only a small set of singular values that are dominant indicating the low-rankness of **H**. This low-rankness justifies our method. However, not all sampling schemes facilitate accurate completion of **H** to predict unknown entries of **H** using MC techniques from only partial measurements.

6.9.2 Process

All the networks selected are directed with unweighted edges. The proposed topology retrieval method is tested on the three representative real-world directed networks in following steps:

1. Sample the network under observation to get **P**

2. Apply low-rank matrix completion on **P** to predict the unknown entries

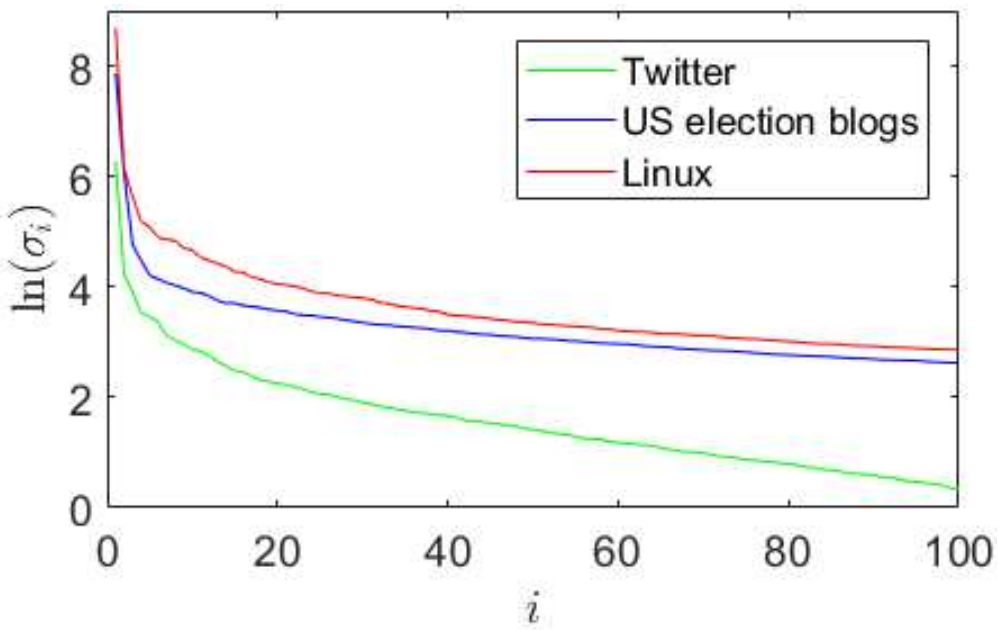


Figure 6.18: Log plot of largest 100 singular values $[\ln(\lambda_i)]$ of **H** for three test networks, namely, Twitter, US election blogs, and Linux files. These plots indicate that the distance matrices of the given directed social networks are naturally close to low-rank.

3. Evaluate the performance, i.e., fidelity with which the network constructed based on sampling represents the original network represented by **H**, using appropriate metrics

Proposed anchor-based sampling is evaluated for 20%, 40%, 60%, 80%, and 90% of nodes as anchors. For random sampling, we vary the percentage of pairwise distances measured, i.e., percentage of known entries of **H**. All results have been averaged over 100 iterations to remove the effects of random seed selection. Next, we explain the parameters used to evaluate the accuracy. All the codes for directed networks can be found in Github at: code-link.

6.9.3 Performance parameters

We use four performance parameters namely - distance distribution, degree distribution, mean error, and absolute hop-distance error to evaluate the accuracy of the proposed approach.

6.9.3.1 Distribution of hop-distances

Distribution of pairwise shortest distance among node pairs provides insight on network connectivity. Accordingly, we can analyze the number of hops required to reach most of the nodes in

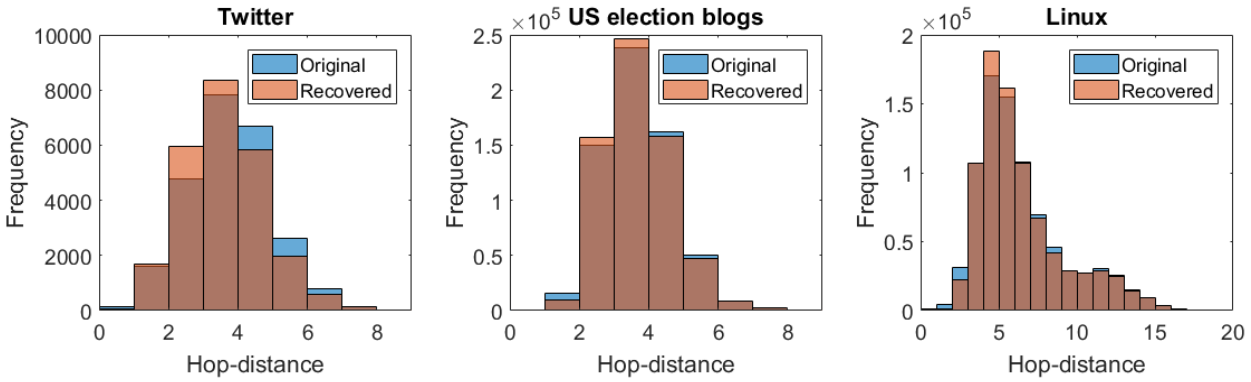


Figure 6.19: Comparison of hop-distance distributions for original (blue) and recovered (orange) networks with 60% of nodes as anchors for Twitter, US election blogs, and Linux networks. The overlap is shown in brown color. Large overlap indicates that we are able to recover the original distributions to a great extent.

the network. As we can see from Fig. 6.19, all the three networks mostly show distributions with high frequency around 4 or 5, implying that a node can reach another within 4 to 5 hops in each network, reducing the communication complexity to a great extent. Fig. 6.19, presents a comparison of distributions for shortest hop-distances in original and recovered networks when only 60% of nodes were selected as anchors under the anchor based sampling scheme. We can see that LMC based recovery is able to preserve the distribution as well as the curve very closely. This implies that the network was well recovered using matrix completion w.r.t. the distance distribution over the network.

6.9.3.2 Distribution of node degrees

Most networks in the real-world have highly skewed degree distributions, meaning that a large majority of nodes have a low degree but a small number, known as "hub-nodes", have high degree [200]. As seen in Fig. 6.20, this is also the nature of the test networks. Thus, it indicates the proportion of hub nodes and other trivial nodes in the network. Fig. 6.20, presents a comparison of degree distributions for original and recovered networks when only 60% of nodes were selected as anchors under the anchor-based sampling. We can see that LMC was able to preserve not only the skewed nature and the long tail of the distribution but also the general shape very closely. Consequently, preserving the proportion of hub and non-hub nodes in the network.

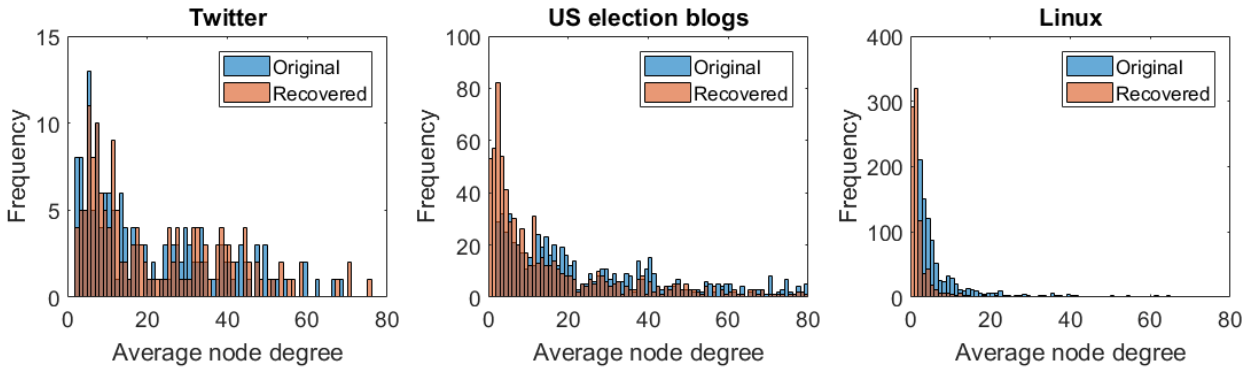


Figure 6.20: Comparison of average node degree distributions for original (blue) and recovered (orange) networks with 60% of nodes as anchors for Twitter, US election blogs, and Linux networks. The overlap is indicated in brown color. Large overlap indicates that we are able to recover the original distributions to a great extent.

6.9.3.3 Mean error

We define mean error M_e as follows:

$$M_e = \left[\sum_{i,j=1}^{N,N} |h_{ij}(f) - h_{ij}|\right] \Big/ \left[\sum_{i=1}^{N}\sum_{j=1}^{N} h_{ij}\right] \tag{6.33}$$

where, $h_{ij}(f)$ refers to the shortest hop-distance from node i to node j in the recovered network when f percentage of random anchors are selected and h_{ij} represents the original shortest hop-distance between nodes i and j. In the case of random sampling, f refers to the percentage of hop-distance matrix elements measured.

Note, that the mean error is relative to the original hop measurements in the network and the significance of an error value might vary for different networks depending on their average shortest hop-distances. Fig. 7.8 presents measured mean error for the two sampling schemes for the three test networks. Anchor-based sampling scheme produces significantly better results compared to random sampling with less than 7% deviation relative to the original measurements for 60% of nodes as anchors. The error value approaches almost 0% as we increase the percentage of sampled data. Error values are relatively higher for the random sampling scheme as we have less relative information about the missing entries.

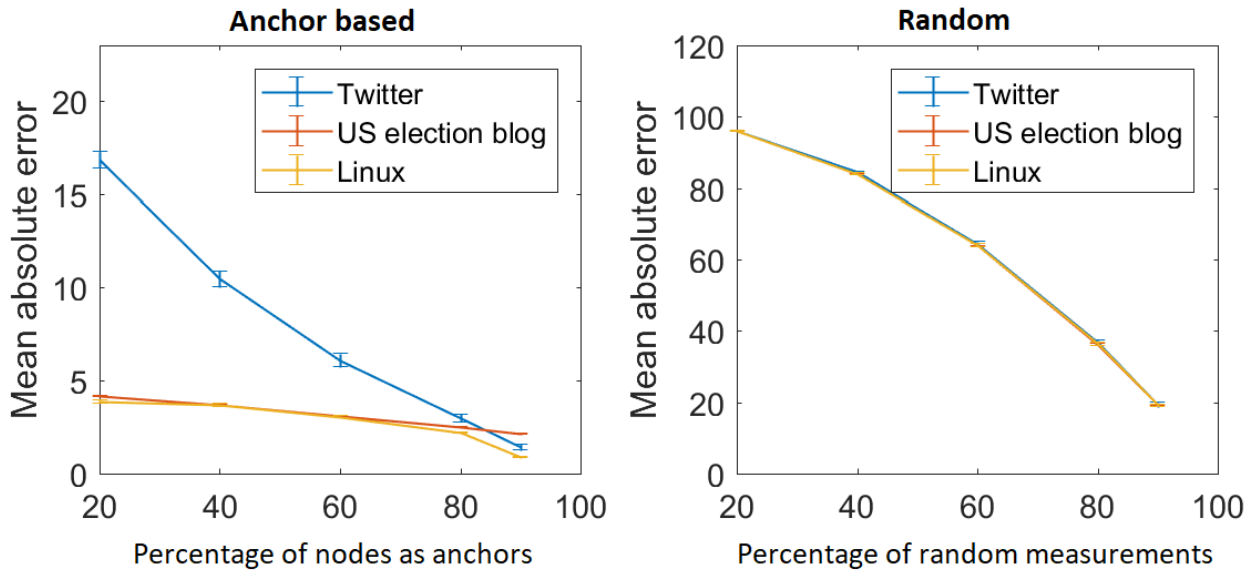


Figure 6.21: Mean error versus percentage of sampling for Twitter, US election blogs and Linux networks under proposed sampling schemes. Anchor based scheme performs better than the random sampling scheme and prediction performance is better for larger networks as at a certain percentage, more number of node pair distances are being sampled in a larger network.

6.9.3.4 Absolute hop-distance error

Absolute hop-distance error, captures average deviation in terms of number of hops for a path, averaged over all possible pair-wise paths for the entire matrix. The absolute hop-distance error is defined as

$$H_e = \left\{ \sum_{i,j=1}^{N,N} |h_{ij}(f) - h_{ij}| \right\} \Big/ \{N \times N\} \tag{6.34}$$

Product $N \times N$ gives the total number of elements in the **H**.

Value of H_e indicates the deviation from the original hop-distance measurement. For instance, if the absolute hop-distance error 1, implies that on average, a predicted path length will be off by 1 hop.

Fig. 7.10 presents H_e for different sampling schemes for the three test networks. We can see that anchor-based sampling scheme produces the best results with less than 0.3 units of deviation relative to the original measurements. The error approaches very close to 0 as we increase the percentage of sampled data.

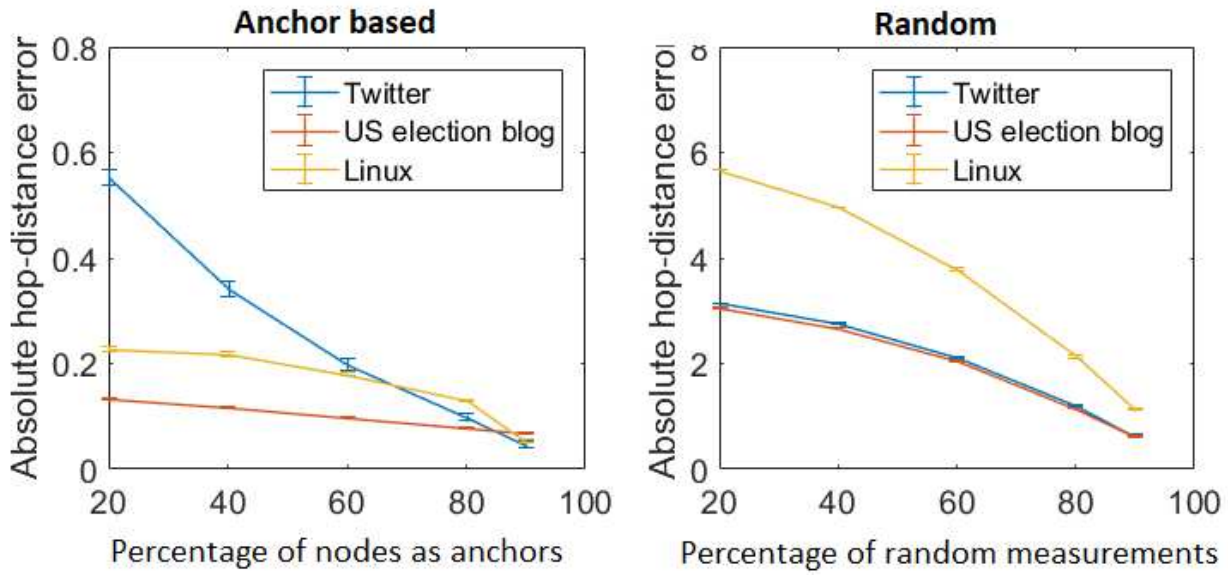


Figure 6.22: Absolute hop-distance error versus percentage of sampling for Twitter, US election blogs and Linux networks under proposed sampling schemes. Anchor based scheme performs better than the random sampling scheme and prediction performance is better for larger networks as at a certain percentage, more number of node pair distances are being sampled in a larger network.

Results show that the sampling and recovery schemes are able to retain basic network features while recovering the network topology from only a fraction of path measurements. Though random sampling works for undirected graphs [87], for directed graphs, we see that anchor based sampling outperforms random sampling with much better results in terms of smaller error values and thus can be used for real-world large networks for recovering network topology from partial measurements.

6.10 Summary for Directed Network Recovery

A low complexity technique to capture characteristics of directed networks such as social networks has been presented. It samples the network using a relatively small set of pairwise distances. The low-rank nature of real-life social networks (Twitter and US election blog), and a large Linux file system network, was demonstrated. These network structures have evolved under different conditions and time scales, yet, they exhibit a low-rank distance matrix structure, which has hitherto been recognized only with respect to large undirected networks. We propose a low-rank matrix completion based technique to recover missing connectivity information from directed social net-

works. Results demonstrate the accuracy and effectiveness of the approach in recovering network features of directed real-world networks such as distances and degree distribution. Our work and results in the directed domain supports the findings of work done in the undirected domain in the first half of this chapter. As directed networks produce asymmetrical distance matrices, predicting missing node-pair distances in directed networks is a harder problem than the undirected case.

The applications of the proposed technique extend beyond network sampling for recovery. It can, for example, be viewed as a compressed representation for large networks. Thus, it is useful for storing massive network structures and distributing such information, especially in distributed systems. There are many situations in which a network is not completely accessible for measurements, and the proposed method can compensate for such missing data as well.

Though the focus of this work was on social networks, the algorithm is applicable for many other directed networks, such as recommender systems, transportation networks, data dependency graphs, and protein interaction networks to name a few. The proposed approach can be tested on a broader set of directed networks to evaluate the extensibility of the technique. Application specific sampling schemes need to be developed for applications where anchor-based or random sampling is not possible. The proposed method can also be tested with several sampling schemes in the future. Extension of the method to dynamic networks can also be investigated.

Chapter 7

Predicting Distances in Social Networks Using Pre-trained Autoencoders

Analysis of large-scale networks is hampered by limited data as complete network measurements are expensive or impossible to collect. We present an autoencoder based technique paired with pre-training, to predict missing topology information in ultra-sparsely sampled social networks. Randomly generated variations of Barabási-Albert and power law cluster graphs are used to pre-train a Hadamard Autoencoder. Pre-trained neural network is then used to infer distances in social networks where only a very small fraction of intra-node distances are available. Model is evaluated on variations of Barabási-Albert and Powerlaw cluster graphs as well as on a real-world Facebook network. Results are compared with a deterministic Low-rank Matrix Completion (LMC) method as well as an autoencoder trained on partially observed data from the test-network. Results show that pre-trained autoencoder far outperforms LMC when the number of distance samples available is less than 1%, while being competitive for a higher fraction of samples.

7.1 Introduction

Network analytics involving topology and connectivity is valuable in extracting crucial insights into real-world networks such as infrastructure, social networks, crime networks, protein-protein interactions, and social movements to explain the social structure, human behavior, predict crimes, predict drug properties, and estimate the spread of contagious diseases [201]. However, collecting sufficient data for network analysis is not always feasible due to inaccessibility to private nodes, expensive storage, or computational requirements. In the case of very large networks, it is essential to be able to do analytics based on a small amount of data. However, as shown in [20], missing data can have a significant impact on the ability to estimate structural properties (e.g., clustering coefficient, average path length) of a social network. Missing or incomplete data changes the

representative network significantly to affect its original network or node feature values making it difficult to draw meaningful conclusions from given network data.

Statistical network sampling techniques may lead to biased data, questioning the generalization of the results. For example in snowball [18] and random walk [202] sampling schemes, users with a larger number of friends have more control over the sampling process making weaker ties appear of lesser influence whereas the fact can be exactly opposite [19]. Sampling in the form of Ego networks which consist of a focal node (“ego"), nodes it is directly connected to (these are called “alters"), and the ties, if any, among the alters, can be oversimplified versions of the original network and thus, may not support community study or network structure as a whole [203]. Effects of biased network sampling on key network statistics are presented in [204]. Thus, estimating the missing entries corresponding to the original network bridges this gap and makes it possible to apply conventional techniques for network analytics.

Reconstruction of networks from a small set of measurements has been demonstrated with Low-rank Matrix Completion (LMC). LMC has been used for predicting missing data when only partial network measurements are available [44, 87, 205]. Graph based autoencoders are used in [72] to predict missing ratings from partially known users and rating matrix. However, the issue with the existing prediction techniques is that they require a significant amount of data from the network being sampled.

This research aims to predict the network topology from an extremely small amount of measurements, e.g., less than 10% or even 1% of the total pair-wise distances. Our approach is based on a neural network that is pre-trained using information from network models that can be easily generated, such as Powerlaw networks. Specifically, we use Barabási-Albert and Powerlaw cluster models. We use the Robust Hadamard Autoencoder (RHA) model [90] but with modifications for imputing missing values in the hop-distance matrices of networks. RHA has demonstrated usefulness in missing value imputation in the presence of noise, random blocks of missing data, in applications such as image inpainting, and materials processing. The term “Robust” is used in RHA due to the robustness of the model in presence of random noise for the task of image denois-

ing and inpainting. However, in this work, since we use the RHA model only for missing value imputation of network data and do not have any form of noise in the measurements, we modify the objective function and rename it as Hadamard Autoencoder (HA).

In this work, we utilize the HA, randomize the domain parameters of networks that we train on, and then apply the learnings to test new and broader classes of networks that were not used for training. Generally, it is meaningful to use a pre-trained network if both tasks or datasets have something in common. Social networks have common domain characteristics with networks such as Powerlaw networks, thus, allowing us to pre-train these Artificial Neural Networks with ample amount of model based data. Pre-training is especially useful when data in the target domain is inaccessible, scarcely available, complex, or expensive, [206] such as collecting real-world network data.

Here, we randomize domain parameters, such as number of edges and probability of adding a triangle, for training networks so that the pre-trained model will treat the test model as just another variation. Thus, the HA model is used to impute the missing values in the hop-distance matrix of a social network that the model was not trained on. This is especially helpful when the complete network measurements are not accessible, such as criminal networks where the hidden nature of crime leads to missing data points [207].

7.2 Contribution

In this paper, we address the problems due to inadequate network measurements for network data analysis. Our focus is on ultra-low levels of sampling, where only a very small fraction (of the order 1% or less) of distance or connectivity information is available. Our approach relies on predicting missing node-pair distances for the given network using machine learning. Specifically, we adopt a pre-training approach that learns from other randomly generated networks and then predicts the missing node-pair distances for an ultra-sparsely sampled network that the neural network has not been trained on. By selecting appropriate family(ies) of network models similar

to the test networks in nature, learning from simulated networks allows the autoencoder to know a lot about the test network.

The main benefits of our recovery technique are as follows:

1. *Works with ultra-sparse data*: Observations from the actual network under test are not required for training. We train the HA using variations of two randomly simulated networks created by Barabási-Albert and Powerlaw cluster models which can be easily generated on the desired scale.

2. *Real-world predictions*: We utilize the trained HA model to predict unobserved network measurements for real-world Facebook network to illustrate the effectiveness of our technique. The results show how the HA model can be efficiently deployed in real-world where data are scarce and expensive to collect.

3. *Result comparison*: We compare our results against a deterministic LMC method which has performed very well in recovering network characteristics using only partial measurements of above 20% sampled distances. We also compare our results against a neural network that is trained purely on partial measurements from the test network to show the significance of training on domain randomized networks.

4. *Extended application*: The pre-training phenomenon extends the applicability of the presented technique beyond the networks under experimentation in this paper and thus HA can be used to predict missing data from other network types by training the model on corresponding variations of the test network.

Prediction using a pre-trained HA model allows us to perform network analysis on scarce, complicated, and rare networks.

We introduce our problem statement and the contribution of our work in Section I. Highlights of the contribution and benefits of this work are listed in Section II. The required theory and utilized approach for the presented imputation technique is explained in Section III. Section IV presents results and lastly, the conclusion and future work are stated in Section V.

7.3 Theory and Approach

The required concepts behind used recovery methods such as network representation, sampling techniques, the importance of low-rankness in social networks for this work, selection of networks for training the neural network model, and explanation of the two recovery techniques compared are defined and explained in this Section. As this is an extension of our previous work [205], in this Section, we closely follow the concepts and notations to explain the foundation and background of the approach.

7.3.1 Network representation

This work is focused on *connected, unweighted, and undirected graphs*. Such a graph G can be defined by $G = \{V, E\}$, where V is the set of nodes or vertices and E is a set of undirected edges. In a social network, users represent nodes and the edges correspond to communication links or friendships in a social network. Such a graph may be represented by an *adjacency matrix* **A** [143], where

$$\mathbf{A} = [a_{ji} = a_{ij} = 1 \text{ if } (i,j) \in E,\ 0 \text{ otherwise }]. \tag{7.1}$$

We mainly sample *Hop-Distance* Matrices (HDMs), $\mathbf{H} \in \mathbb{N}_0^{N \times N}$, (where $\mathbb{N}_0$ denotes the non-negative integers $\mathbb{N} \cup \{0\}$), with entry h_{ij} corresponding to the shortest path between n_i and n_j.

The ij^{th} entry in **H**, i.e., h_{ij} which locates the i^{th} row and j^{th} column, represents the shortest distance from node n_i to node n_j. **H** is not affected if nodes n_i and n_j have multiple shortest paths. Table 7.1 lists all the notations used in this chapter.

It is known, that **H** is low-rank for many classes of undirected networks [85, 87]. We present results in Section 7.3.3 to validate low-rankness for train and test networks used in this chapter. Low-rankness of a distance matrix implies a significant dependence of distance vectors, i.e., rows of **H**, on each other. Consequently, low-rank prediction techniques have been used, e.g., to predict randomly missing distance entries [85, 87, 205].

Table 7.1. Notation

Notation	Description
$\mathcal{N}$	Set of nodes
N	Number of nodes, $\lvert\mathcal{N}\rvert = N$
$n_i \in \mathcal{N}$	i-th node (current node)
h_{ij}	Minimum hop-distance between nodes n_i and n_j
$\mathbf{H}$	Hop-distance matrix
$\mathbf{A}$	Adjacency matrix
$\mathbf{P}$	Sampled hop-distance matrix
$\hat{\mathbf{P}}$	Predicted hop-distance matrix
λ_i	i-th singular value
$\ddot{\Omega}$	Set of node pairs (i, j) with unobserved hop-distances

7.3.2 Sampling scheme for social networks

We aim to capture the network information accurately using efficient sampling of the hop-distance matrices. A random sampling scheme is used where we measure hop-distances among a very small number of random node pairs. The sampling scheme gives us a fraction of entries from **H**, which can be represented by entries in an $N \times N$ matrix **P** as shown in Fig. 8.2. The sampled entries are shown by solid squares and '?'s represent missing entries. Thus, if the distance from n_i to n_j is sampled, ij^{th} element of matrix **P** is denoted by h_{ij} and remaining missing elements of **P** are denoted by $\mathcal{X}$. After the network is sampled, the rows from **P** can be represented in the form $h_i := [h_{i1}, h_{i2}, \mathcal{X}, .., \mathcal{X}, .., h_{iN}]$. Matrix Completion based approach requires a sampling scheme to measure at least one non-zero distance value in each row for the test network [87]. However, we do not have this restriction for Hadamard Autoencoders.

Random walk based sampling schemes work well for scale-down approaches [197]. However, being biased towards higher degree nodes, random walk produces a misrepresentation of networks with non-uniform degree distribution. This causes a significant practical challenge since typical large-scale, real-world, unstructured networks tend to have non-uniform degree distributions, e.g., unstructured peer-to-peer networks exhibit power law degree distribution [198]. Thus, we use random measurements throughout the network for a distributed unbiased sampling of the network.

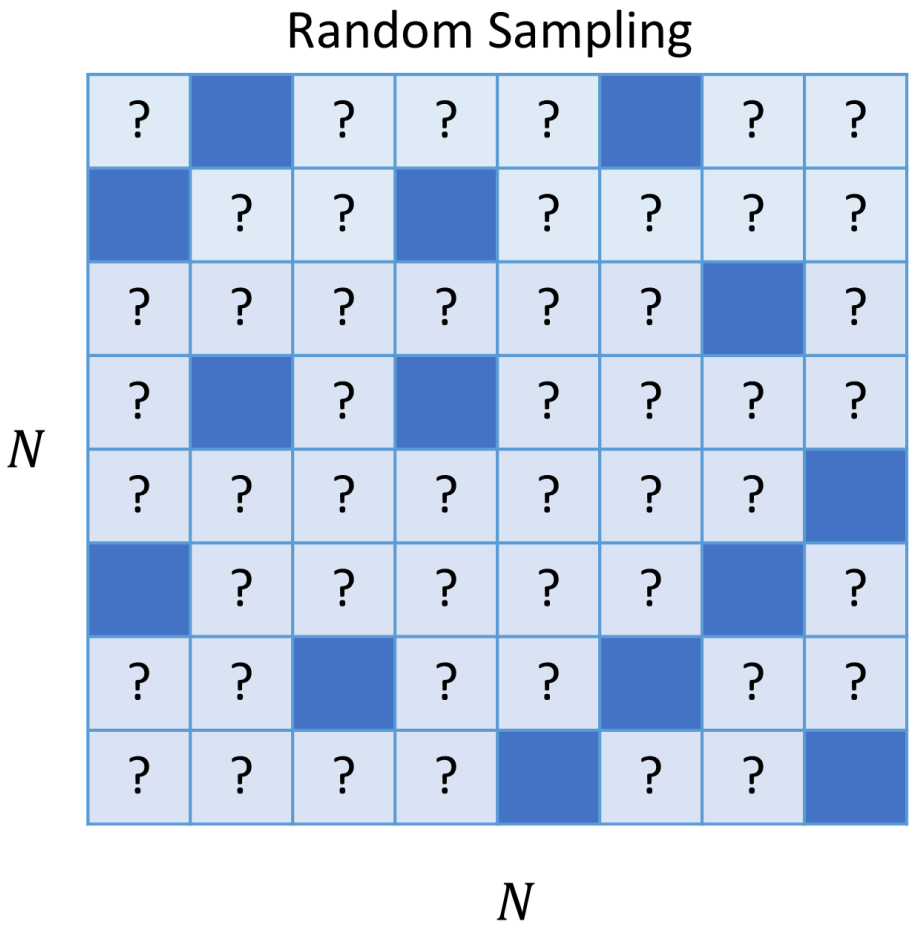


Figure 7.1: Visual illustrations of **P** (sampled **H**) for random sampling scheme for $N = 8$. Known entries are denoted by dark blue and unknown values by light blue.

7.3.3 Low-rankness in social networks

Here, we demonstrate the low-rankness of the training and testing networks. Fig. 7.2 shows the logarithm of all the singular values of **H**. Only a small set of singular values are dominant and the later magnitudes fall rapidly towards '0' indicating low-rankness of the plotted **H** matrices for shown networks. This low-rankness justifies our recovery method. However, every sampling schemes may not facilitate accurate completion of **H** to predict the unknown entries from only partial **H**. Note that, low-rank data lies on low-dimensional linear subspace whereas low-dimensional data lies on low-dimensional non-linear subspace. The LMC method worked well for networks sampled above 20% of the total node-pair distances [87].

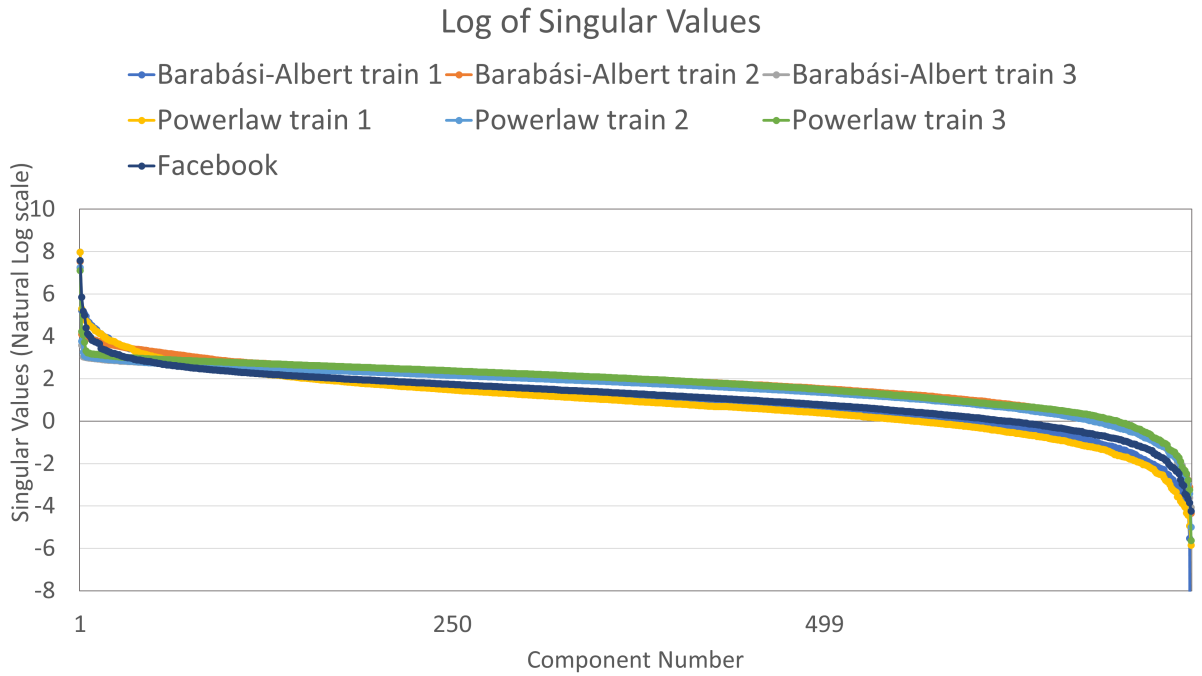


Figure 7.2: Log plot of singular values of **H** for Barabási-Albert, different Powerlaw networks, and Facebook test networks. These plots indicate that they are naturally close to low-rank.

7.3.4 Network Selection

The empirical evidence collected from the analysis of both natural and human-created networks from the real-world have shown the presence of power law behavior in their degree distributions, i.e., the fraction $P(k)$ of nodes in the network having k connections to other nodes goes for large values of k as $P(k) \sim k^{-\xi}$, with the exponent varying between 2 and 3. The characteristic that only a few nodes have an extremely large number of connections while most of the nodes have very few links is caused due to preferential attachment [208, 209]. Furthermore, social networks like Facebook and WeChat are observed to have power law growth dynamics [210].

We intend to make our model work on networks with preferential attachment because several real-world networks such as social networks are approximately scale-free. Thus, their degree distributions follow a power law, at least asymptotically. Such networks have fewer nodes with very high node degrees and a large number of nodes with only few neighbors [211]. While the networks are evolving in size at a very high rate, the sampling techniques and large-scale processing algo-

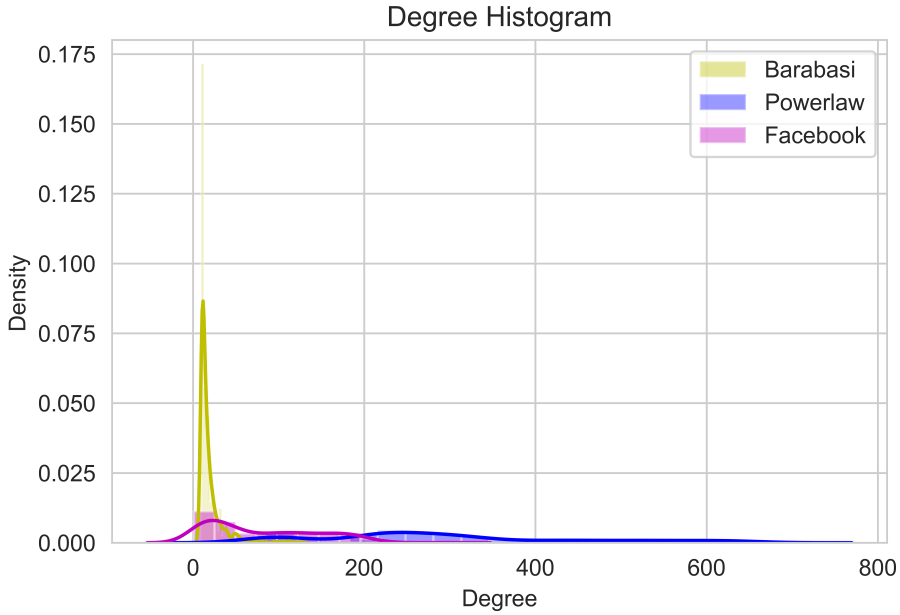


Figure 7.3: Node degree distributions against density for one of the variations of Barabási-Albert (BA), Powerlaw cluster networks, and the Facebook test network. The range of distributions in the training networks help the neural network adapt for a test network within the given range.

rithms are not. So, to solve the evident problem of insufficient data for network analysis, being able to predict missing node-pair distances on such networks gives a huge advantage to numerous areas such as biology, the entertainment industry, criminology, computer science, communications, etc.

The Barabási-Albert (BA) and Powerlaw cluster models create graphs for the given network size and edges using preferential attachment where a graph of N nodes is grown by attaching new nodes each with m edges that are preferentially attached to existing nodes with high degree. Both the models generate random networks whose degree distributions follow a power law curve and thus, represent networks such as protein interaction, the Internet, rating networks, communication networks, World Wide Web, citations network, and Facebook network.

Powerlaw cluster graphs are essentially the Barabási–Albert (BA) growth model with an extra step that each random edge is followed by a chance of making an edge to one of its neighbors too (and thus a triangle). This algorithm improves on the BA model in the sense that it enables a higher average clustering to be attained if desired. Thus, it helps represent networks with higher clustering

coefficients than BA graphs with similar network size and edges per node [212]. Selecting these two types of networks with preferential attachment helps us train the neural network to predict distances in social networks with preferential attachment. Fig. 7.3 shows the difference in the degree distribution spreads for a Barabási–Albert (BA) and a Powerlaw cluster graph despite their commonality which fosters the adaptability of our technique towards variations in network types in testing.

7.3.5 Network recovery techniques

With the proposed random sampling technique, some entries in **P** are *not observed* and therefore the matrix **P** is incomplete. In this Section, we study how HA can be used to recover the missing node pair measurements in a scale-free social network. We also study the Low-rank Matrix Completion method in brief against which the performance of the HA model is compared.

7.3.5.1 Hadamard Autoencoders

We use the HA model as given by [90, 213] for imputing missing values in the hop-distance matrix (HDM) of two types of random graph networks. The objective function used for this task is given below:

$$\min_{\theta} ||(\mathbf{X} - D(E(\mathbf{X}))) \odot \Omega||_2 \tag{7.2}$$

Here, **X** is the incomplete distance matrix that is used as input to the neural network. The unobserved entries in the matrix are replaced with zeros before training.

Fig. 7.4 visualizes the concept of the HA model. The $\odot$ represents the Hadamard or element-wise product between entries of the cost function and Ω. Ω is an indicator matrix of ones and zeros. Every unobserved entry in the dataset has a corresponding entry of zero in the indicator matrix whereas, every entry in the dataset that is observed has a corresponding entry of one in the indicator matrix. Thus, the objective function ignores the regions in the data that are missing and the optimal values for imputation are found based on the surrounding values that are observed

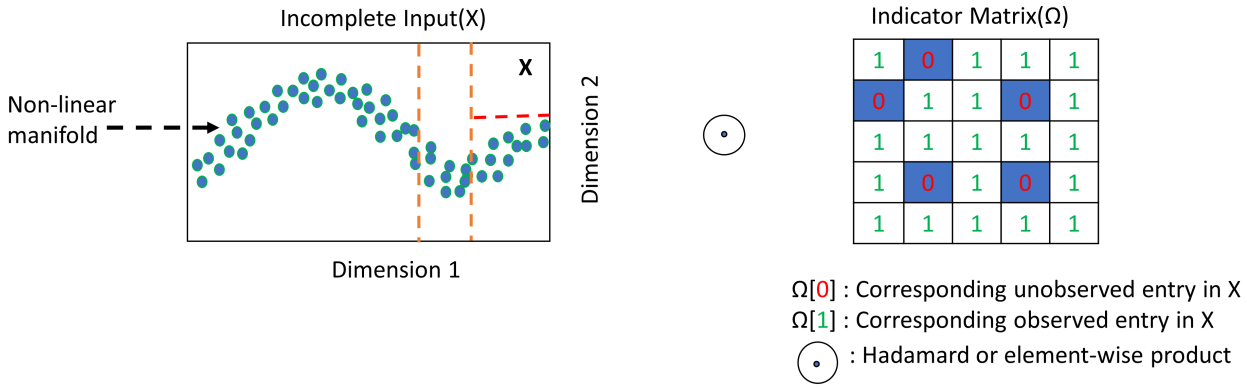


Figure 7.4: Visualization of HA concept. The autoencoder learns a non-linear manifold on which the data lay as shown in the 2D plot. We have incomplete input **X** with random entries missing. We multiply the input element-wise with an indicator matrix. The indicator matrix is defined as a matrix of ones and zeros where the missing entries in **X** have corresponding zeros in the indicator matrix and the observed entries in **X** have corresponding ones in the indicator matrix. During training, the cost function will ignore the entries that are zeros in the input, and the model will be optimized only based on the observed entries. The missing values are thus imputed based on the surrounding observed values.

during training. For training the HA model, we performed hyper-parameter tuning to find the best combination of parameters in terms of both metrics, i.e., mean error and Absolute hop-distance error. In particular, we used one hidden layer with leaky_relu as the activation function. The learning rate used for all experiments was 0.01 with a batch size of one. The total number of training epochs for each of the graphs in our training set was 10. The predictions were made on graphs that were not seen during training in order to assess how well the model generalizes on new graphs. In our first experiment, we trained the HA model on three Powerlaw cluster graphs and tested it on a different variation of the Powerlaw cluster graph. In the second experiment, we trained the model on three Barabási-Albert graphs and tested on a new variation of the Barabási-Albert graph. The specifications of training and testing networks are as given in Table 7.2.

7.3.5.2 Low-rank Matrix Completion

Herein, one of our key assumptions is that the networks under analysis are low-rank, and therefore we have leveraged ideas from low-rank *matrix completion* [51, 123].

The main concept of such methods can be presented as the following optimization problem:

$$\hat{\mathbf{P}} = \arg\min_{\mathbf{P}_0} \rho(\mathbf{P}_0), \;\; \text{s.t. } \mathcal{P}_\Omega(\mathbf{P}) = \mathcal{P}_\Omega(\mathbf{P}_0), \tag{7.3}$$

where ρ returns the rank of a matrix, $\mathbf{P}$ is a partially observed matrix, $\hat{\mathbf{P}}$ is the desired complete $\mathbf{P}$ matrix with all the missing entries filled in. In addition, Ω denotes a set of observed entries of the matrix $\mathbf{P}$ and $\mathcal{P}_\Omega$ denotes a projection of $\mathbf{P}$ onto Ω. Accordingly, the constraint above imposes that $\mathbf{P}_0 = \mathbf{P}$ for the observed entries of $\mathbf{P}_0$. The main idea is that we try to find a matrix $\mathbf{P}_0$ such that the rank of $\mathbf{P}_0$ is minimized while imposing that the observed entries are preserved in the constructed matrix, i.e., $\mathcal{P}_\Omega(\mathbf{P}) = \mathcal{P}_\Omega(\mathbf{P}_0)$. Thus, the returned matrix $\mathbf{P}_0$ will be faithful to our measured hop-distances; but, $\mathbf{P}_0$ is free to take any values outside of Ω to minimize its rank. Unfortunately, Eq. (7.3) is an exponentially expensive optimization problem [51, 123] that can only be solved for trivial networks. However, the problem in Eq. (7.3) can be recast [51, 123] as a convex optimization problem as given below:

$$\hat{\mathbf{P}} = \arg\min_{\mathbf{P}_0} \|\mathbf{P}_0\|_*, \;\; \text{s.t. } \mathcal{P}_\Omega(\mathbf{P}) = \mathcal{P}_\Omega(\mathbf{P}_0), \tag{7.4}$$

which can be solved efficiently for millions of nodes on commodity computing hardware using splitting techniques and iterative matrix decomposition algorithms [121, 123]. In Eq. (7.4) we have exchanged the rank ρ for the *nuclear-norm* $\|\mathbf{P}_0\|_*$, which is the sum of the singular values of $\mathbf{P}_0$ which leads to a *convex* optimization problem [121, 123]. Even though a matrix of distances such as $\mathbf{P}$ should not have negative entries, the reconstructed matrix $\hat{\mathbf{P}}$ might have such entries. As LMC is not explicitly designed to handle such problems, we round-up all the negative values of $\hat{\mathbf{P}}$ to zero and positive values to the closest positive integer. There are many standard libraries for solving convex optimization problems such as Eq. (7.4). Herein, we have leveraged the library CVXPY [157] in the scripting language Python [158] to solve the problem in Eq. (7.4) using the pseudo-code as described in Algorithm 9. There have been several research solutions on recovering missing entries of Euclidean distance matrices in undirected as well as directed graphs [51], however, to the best of our knowledge, recovering network data without using any observed entries

Algorithm 9 Compute low-rank representation $\hat{\mathbf{P}}$ of the partially observed HDM $\mathbf{P}$.

Require: Partially complete hop-distance matrix $\mathbf{P}$.

1: **procedure** MATRIX COMPLETION
2: Compute recovered matrix $\hat{\mathbf{P}}$ using 7.4
3: Round-up all diagonal values of $\hat{\mathbf{P}}$ to 0 and off-diagonal values between 0 and 1 to 1
4: **end procedure**

from the test network is a novel technique and has not been done for undirected graphs such as scale-free networks.

7.4 Results

In this Section, we present results through two experiments described as follows:

1. *Experiment 1*: In the first experiment, we train the HA model using three variations of Barabási-Albert and Powerlaw cluster networks and then test it on a new variation of Barabási-Albert and Powerlaw cluster networks. Specifications of networks used are listed in Table 7.2. Networks *A,B,C,* and *test* are training and testing networks respectively.

2. *Experiment 2*: In the second experiment, we train the HA model using three variations of Barabási-Albert and Powerlaw cluster networks and then test it on a real-world Facebook network with 744 nodes, 30023 edges, and an average node degree of 80. Specifications of networks used are listed in Table 7.2. Networks *1,2,3,* and *test* are training and testing networks respectively. Note that we do not fine-tune the network with observed measurements from the test network.

The model performance is compared with the Matrix Completion method for a state-of-the-art comparison of the results. We also compare our results against three trivial cases as described below:

1. *Trivial replaced with 0*: The performance metric is calculated against a dummy result. This is obtained by setting all the diagonal entries in the Facebook network to '0' and replacing all the missing entries with '0'.

2. *Trivial replaced with 1*: The performance metric is calculated against a dummy result. This is obtained by setting all the diagonal entries in the Facebook network to '0' and replacing all the missing entries with '1' as '1' is the shortest possible distance between two distinct nodes.

3. *No training*: The HA model is initialized to random weights and used as is for predicting the shortest possible distance between two distinct nodes in the Facebook network. The performance metric is calculated against this resultant distance matrix.

Trivial replace with 0 and *Trivial replace with 1* cases help us understand the worst case error when we do not know anything during reconstruction and predict all entries as a constant. The *No training* case helps us understand the worst case error when we do not train the HA model and predict with the randomly initialized weight values.

It is important to note that the *No training* HA does leverage information about the class of networks that we study. In particular, the dimension of the hidden layer is chosen to minimize the prediction errors. Somewhat surprisingly, even this small measure of information substantially improves prediction accuracy in the sparse training data limit.

The Facebook test network has 744 nodes, 30023 edges, and an average node degree of 80. We replace randomly selected values from the distance matrix in the training graphs with 0 and then feed the incomplete matrix as input to the neural network model. We evaluate our model for a small percentage of sampled entries from the complete hop-distance matrix. In this article, we present results for data sampled over 80%, 60%, 40%, 20%, 10%, 1%, 0.5%, 0.1%, and 0.0001% from the complete hop-distance matrix. Note that these values are rounded and may vary by 0.1% in the actual sampled data. Fig. 7.5 presents a distribution of the number of observed entries per row in the HDM for 1% and 0.1% of sampled entries showing ultra-sparse measurements in the sampled network measurements.

In the second experiment, we train the HA model on three Barabási-Albert graphs each with diameter three and test on a fourth variation of a Barabási-Albert graph with diameter three. The specifications of network parameters used during training are given in Table 7.2. The process

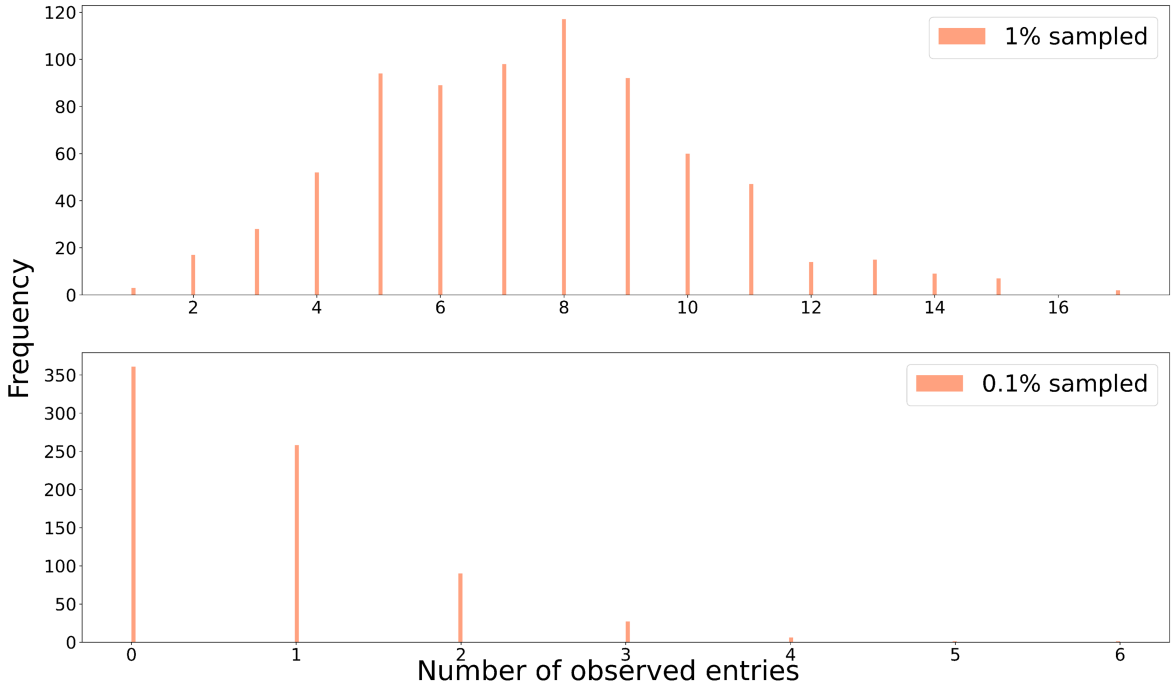


Figure 7.5: Illustration of sparsity in training data: The histograms shows the number of observed entries per row of the training matrix for 1% and 0.1% of total sampled entries.

of replacement of missing values before feeding the incomplete matrix to the neural network is similar to that described in the case of Powerlaw graph networks. We optimize the parameters of the HA model in a way such that it can generalize on new variations of the graphs that were not originally used during training. The details of the parameters used for training the HA model are covered in the section on Network recovery techniques. Hyperparameter tuning is critical while selecting the model configuration such that it does not overfit the training data and generalizes well on networks it has not been trained on.

We use the *networkx* library in Python to generate the Barabási-Albert and Powerlaw clusters [214]. Barabási-Albert network is generated using the function *barabasi_albert_graph(N, m, seed=None)* and Powerlaw cluster network is generated using the function *powerlaw_cluster_graph(N, m, p, seed=None)*, where N is the size of the network, m is the number of random edges to add for each new node, and p is the probability of adding a triangle after adding a random edge.

We train the HA model for values listed in Table 7.2.

Table 7.2. Parameters of Barabási-Albert and Powerlaw cluster networks used in training and testing

Network	N	m	p
Barabási-Albert train A	2000	20	-
Barabási-Albert train B	2000	30	-
Barabási-Albert train C	2000	40	-
Barabási-Albert test	2000	35	-
Powerlaw cluster train A	2000	100	0.6
Powerlaw cluster train B	2000	300	0.7
Powerlaw cluster train C	2000	500	0.8
Powerlaw cluster test	2000	400	0.75
Barabási-Albert train 1	744	2	-
Barabási-Albert train 2	744	10	-
Barabási-Albert train 3	744	50	-
Powerlaw cluster train 1	744	2	0.1
Powerlaw cluster train 2	744	50	0.5
Powerlaw cluster train 3	744	200	0.9

7.4.1 Process

All the selected networks are undirected with unweighted edges.

The proposed method for recovering missing network measurements is tested on representative real-world undirected networks in following steps:

1. Generate three variations of the network under observation
2. Sample the networks used for training to get **P** for all three network variations
3. Replace the missing entries by 0
4. Train the Hadamard Autoencoder with the sampled network measurements
5. Impute missing node-pair distances for Facebook test network
6. Evaluate the performance, i.e., fidelity with which the network is constructed for various sampling percentages, using performance metrics described in Section 7.4.2

Fig. 7.6 shows the setup for training and testing using the Hadamard Autoencoder model. Train test FB results are generated by training the neural network on observed distance measurements from Facebook test network and imputing the missing values. Proposed random sampling is eval-

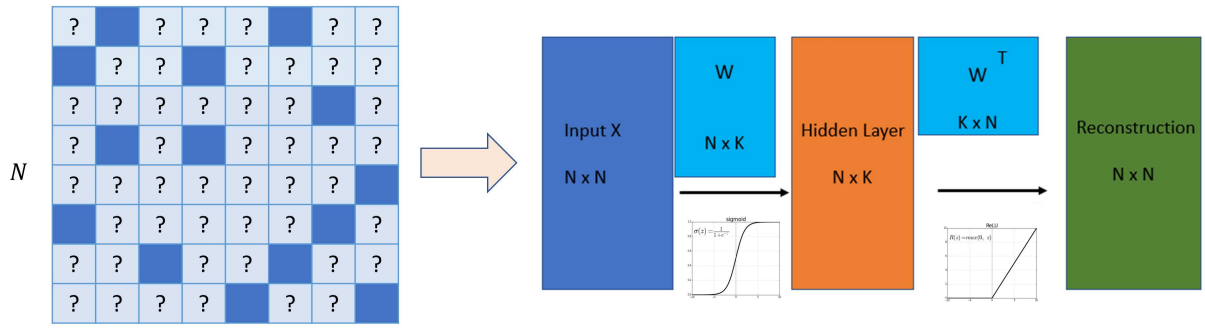


Figure 7.6: Experimental setup for training and testing the Hadamard Autoencoder model. We train on the incomplete distance matrix of either of Powerlaw cluster, Barabási-Albert or Facebook networks. N is the size of the network and k representing a k-dimensional manifold. The missing entries of the training networks are replaced by 0 prior to feeding the incomplete matrix to the autoencoder as input.

uated for data sampled over 80%, 60%, 40%, 20%, 10%, 1%, 0.5%, 0.1%, and 0.0001% from the complete hop-distance matrix **H**. Note that as **H** is a symmetric matrix, for each deletion of entry h_{ij}, entry h_{ji} is also deleted from the sample to make sure that the path length between n_i and n_j is completely deleted. The Python scripts and data used for training and testing Hadamard Autoencoder can be found at [215]. All results have been averaged over 25 iterations to remove the effects of random selection. In the next subsection, we explain the parameters used to evaluate the accuracy of the proposed recovery techniques.

7.4.2 Performance parameters

We measure and compare mean error and absolute hop-distance error to evaluate the accuracy and effectiveness of the pre-trained autoencoder based approach and compare it to Matrix Completion based approach. The errors are measured only over the unobserved node pair measurements. As we know that all diagonal values in a distance matrix are '0' and there are off-diagonal entries which are '0', we set all diagonal values of $\hat{\mathbf{P}}$ to 0 and round-up all off-diagonal values between 0 and 1 to 1.

7.4.2.1 Mean error

We define mean error M_e as follows:

$$M_e = \left[\sum_{\forall\ i,j \in \check{\Omega}} |\check{h_{ij}}(f) - h_{ij}| \right] \times 100 \Big/ \left[\sum_{\forall\ i,j \in \check{\Omega}} h_{ij} \right] \tag{7.5}$$

where, $\check{\Omega}$ is a set of all node pairs whose shortest hop-distances are unobserved, $\check{h_{ij}}(f)$ refers to the shortest hop-distance from node i to node j for unobserved node pair measurements in the recovered network when f percentage of random measurements are unobserved and h_{ij} refers to the original shortest hop-distance between n_i and n_j for unobserved node pairs.

Mean error is a relative error measured with respect to the original hop-distances in the network and is expressed as a percentage. Note that the significance of an error value might vary for different networks depending on their sum of shortest hop-distances. For instance, if their sum of shortest hop-distances is 100 then a mean error of 10% implies that the predicted measurement will be off by 0.1 in magnitude.

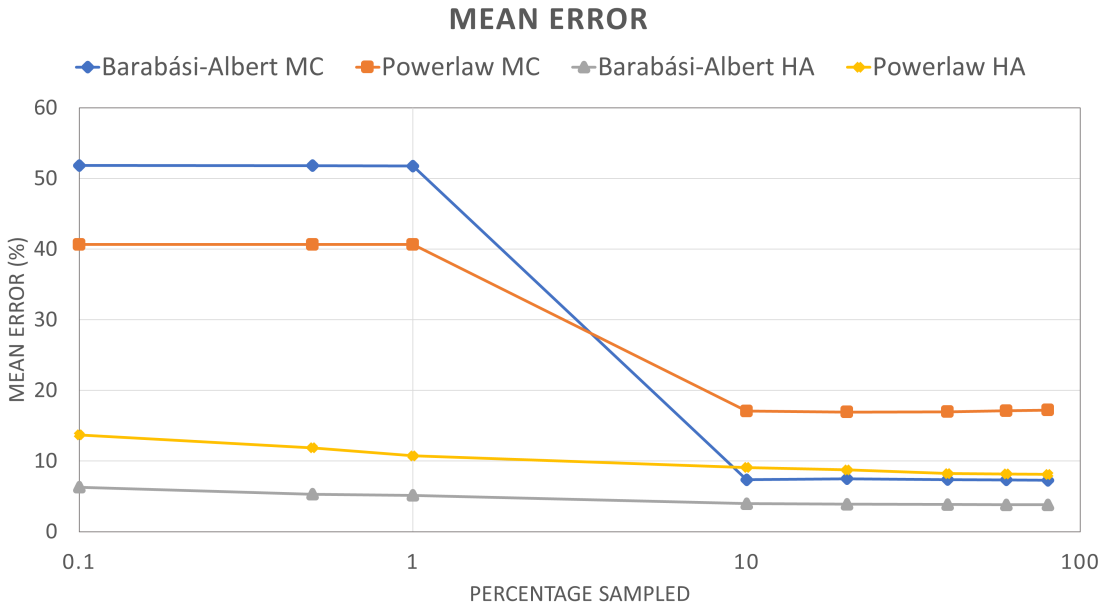


Figure 7.7: Experiment 1 - Mean error (%) results for HA and Low-rank Matrix Completion for various percentages of observed distance entries. HA results outperform LMC results by a huge margin for ultra-sparse measurements.

For the presented Fig. 7.7 and Fig. 7.8, X-axis is the percentage of observed entries from the complete distance matrix of the network. X-axis is presented on a logarithmic scale with base 10. Y-axis is the mean error metric (%) calculated only on the unobserved entries of Facebook. Fig. 7.7 shows comparison of the imputation results for Experiment 1. *Barabási-Albert HA* and *Powerlaw HA* refers to the imputation results of HA, pre-trained on three variations of BA and Powerlaw cluster networks each when the model was tested on BA and Powerlaw cluster graphs respectively. *Barabási-Albert MC* and *Powerlaw MC* refers to imputation results of LMC, when tested on BA and Powerlaw cluster graphs respectively. It can be seen that HA shows a graceful degradation in performance as compared to the LMC method which shows a steep degradation for data sampled below 10%. The HA results exceed LMC results by a huge margin. Even for 0.5% of sampled entries, HA imputations are only 5 to 10% away from the actual distance magnitudes whereas LMC predictions are 40 to 45% away from the actual values.

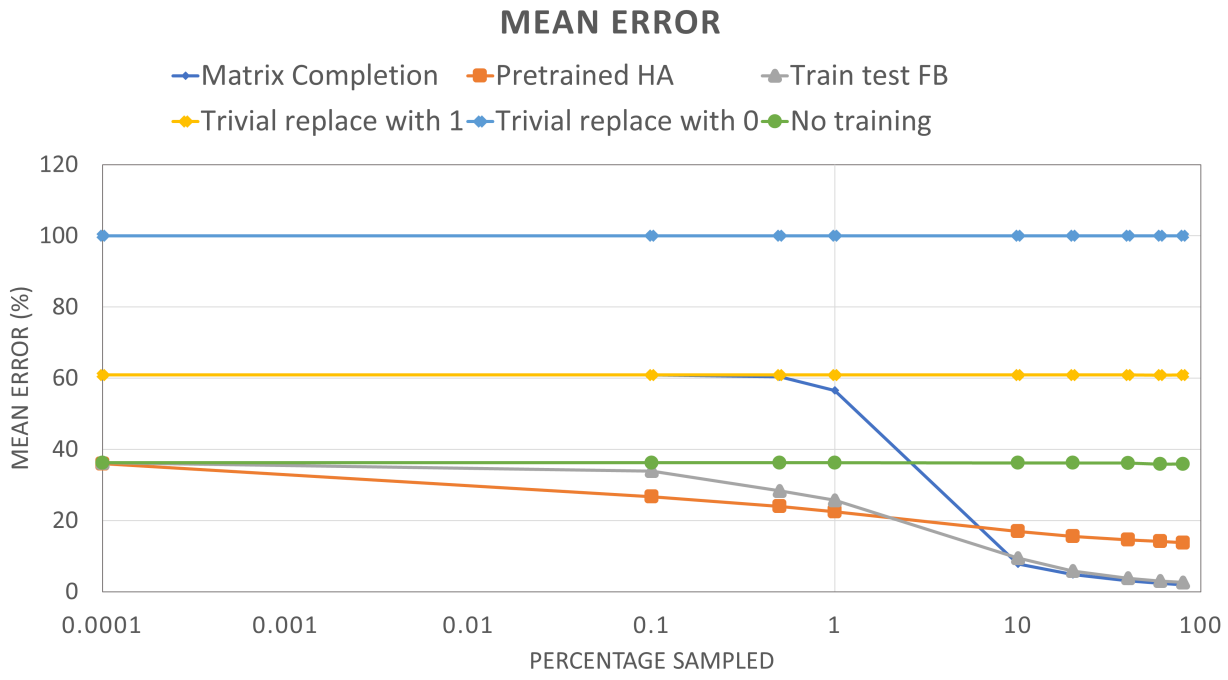


Figure 7.8: Experiment 2 - Mean error (%) results for pre-trained HA, Low-rank Matrix Completion, and three Trivial cases for various percentagess of observed distance entries. HA outperforms LMC results for ultra-sparsely sampled measurements.

Fig. 7.8 presents result comparison for Experiment 2 with the three Trivial cases listed at the beginning of Section III. *Matrix Completion* refers to imputation results of LMC, when tested on Facebook test network. *Pre-trained HA* refers to imputation results of a HA pre-trained on three variations of BA and Powerlaw cluster graphs and tested on Facebook network. *Train test FB* refers to prediction results of HA after training on observed entries of Facebook and testing it on unobserved entries of Facebook. *Trivial replace with 0*, *Trivial replace with 1*, and *No training* are the three Trivial cases.

It can be seen that LMC outperforms the HA model but performs almost equivalently to the train test FB method for all sampled percentages up to the 10% of sampled entries. However, beyond 10% of sampled entries, the neural network exhibits a graceful degradation in performance as compared to the LMC method which shows a steep degradation in terms of the mean error. The pre-trained HA model slightly outperforms the train test FB method at lower fractions of observations. This is because the model has more data to learn from multiple random graphs as compared to learning only from a few observed entries of the Facebook network at higher missing percentages.

Note that LMC matches the worst case results, i.e., *Trivial replace with 1* case error values for lower fractions of sampled measurements, LMC behaves as if there is almost no knowledge of the test data. Similarly, HA results match the worst case results, i.e., *No training* case error values for 0.01% of sampled entries and above as at these high percentage of missing values, the training data is almost absent and thus the HA relies mainly on its initial random weights for imputations. The pre-tained HA still outperforms train test FB results as it has more data to learn from multiple random graphs as compared to learning only from a few observed entries of the Facebook network at lower fractions of sampled data.

7.4.2.2 Absolute hop-distance error

Absolute hop-distance error (AHDE), captures average deviation in terms of hop-distances for the entire matrix. AHDE is defined as

$$H_e = \left\{ \sum_{\forall\ i,j \in \check{\Omega}} |\check{h_{ij}}(f) - h_{ij}| \right\} \Big/ \{card(\check{h})\} \tag{7.6}$$

Where $\check{h}$ represents set of all unobserved entries and $card(\check{h})$ gives the total number of unobserved entries in **H**.

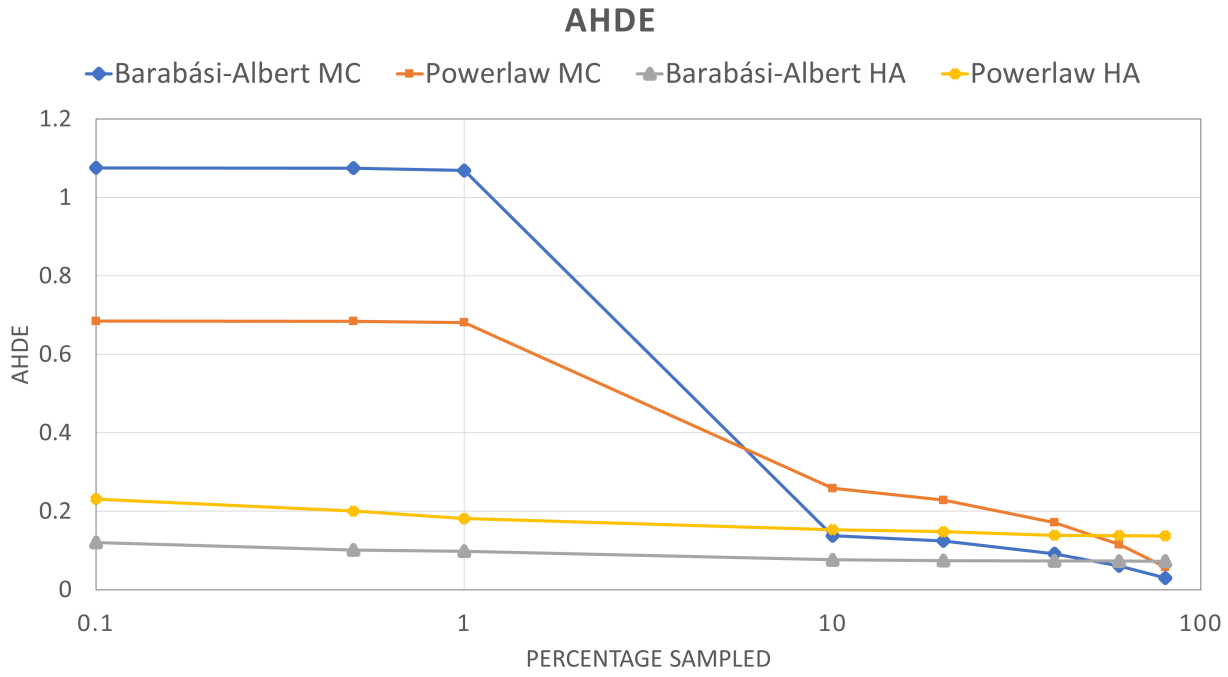


Figure 7.9: Experiment 1 - Absolute hop-distance error results for Hadamard Autoencoder (HA) and Low-rank Matrix Completion for various percentagess of observed distance entries. HA results outperform LMC results by a huge margin for ultra-sparse measurements. HA results outperform LMC results by a huge margin for ultra-sparse measurements.

The value of H_e indicates the deviation of the predicted value from the original hop-distance measurement in magnitude. For instance, the absolute hop-distance error of 1, implies that on average, a predicted path length will be off by 1 hop, i.e., for the original hop-distance of 10, predicted value will be within 9 and 11.

For the presented Fig. 7.9 and Fig. 7.10, X-axis is the percentage of observed entries from the complete distance matrix of the network. X-axis is presented on a logarithmic scale with base 10. Y-axis is the AHDE metric calculated only on the unobserved entries of Facebook. Fig.

7.9 displays result comparison for Experiment 1. *Barabási-Albert HA* and *Powerlaw HA* refers to imputation results of HA, pre-trained on three variations of BA and Powerlaw networks when the model was tested on BA and Powerlaw cluster graphs respectively. *Barabási-Albert MC* and *Powerlaw MC* refers to imputation results of LMC, when tested on BA and Powerlaw cluster graphs respectively. Pre-trained HA outperforms LMC results for ultra-sparse sampling, i.e., on and above 1% of sampled entries. While LMC predicts distance measures with an error margin of 0.6 to 0.9, this may cause the predicted hop-distance value to deviate while rounding off the entries. This is not the case for pre-trained HA as the error margin lies within the magnitude of 0.1 to 0.2, leading to more accurate hop-distance imputations. HA results show a graceful degradation in performance as compared to the *Matrix Completion* which shows a steep degradation below 10% of sampled entries.

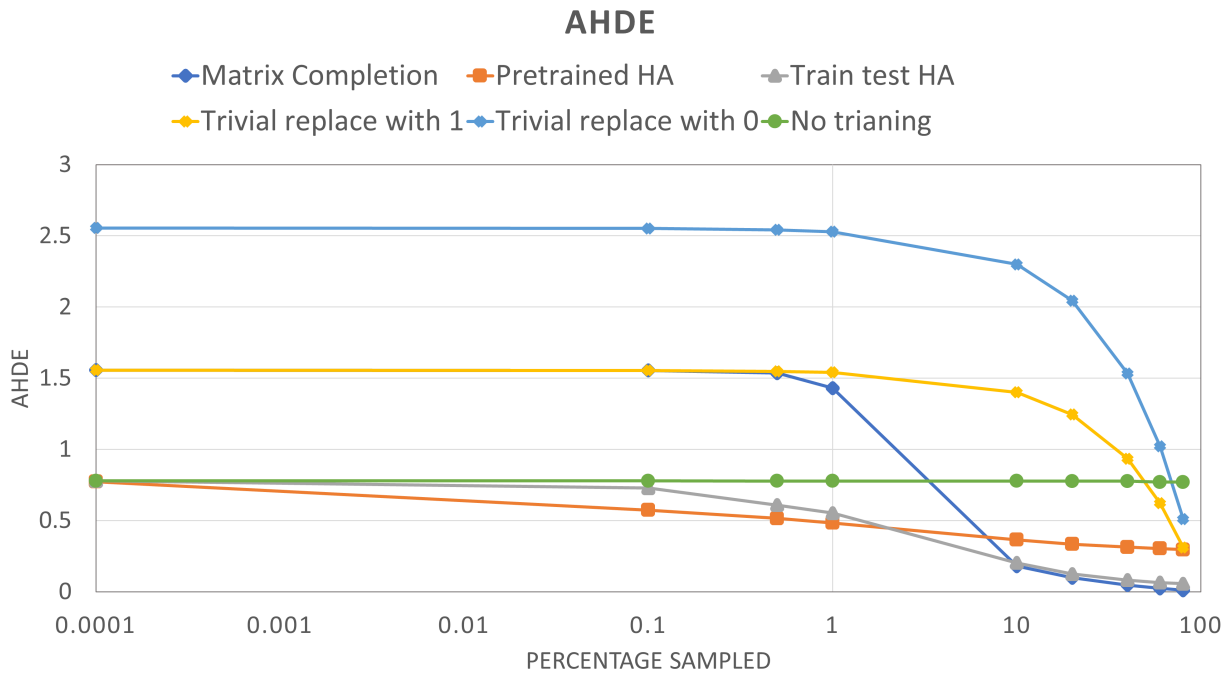


Figure 7.10: Experiment 2 - Absolute hop-distance error results for pre-trained HA, Low-rank Matrix Completion, and three Trivial cases for various percentages of observed distance entries. HA outperforms LMC results for ultra-sparsely sampled measurements.

Fig. 7.10 presents results for Experiment 2 in comparison with three Trivial cases listed at the beginning of Section III. *Matrix Completion* refers to imputation results of LMC when tested on Facebook test network. *Pretrained HA* refers to imputation results of a HA pre-trained on three variations of BA and Powerlaw cluster graphs and tested on the Facebook network. *Train test FB* refers to prediction results of HA after training on observed entries of the Facebook and testing it on unobserved entries of Facebook. *Trivial replace with 0*, *Trivial replace with 1*, and *No training* are the three Trivial cases.

Pre-trained HA slightly outperforms the train test FB method for sampling below 10% similar to that seen for mean error. This is because there is more data to learn from for the former case as compared to the latter at higher missing percentages. It can be seen that at higher sampled percentages, i.e., up to 10% of sampled entries, the train test FB and matrix completion methods show comparable performance, similar to that seen for the mean error metric. Beyond the 10% of sampled entries, however, the neural network achieves a graceful degradation in performance as compared to the matrix completion method beyond that.

Similar to mean error results, the worst case performance of LMC matches the *Trivial replace with 1* case and the worst case performance of HA matches the error values from *No training* case due to lack of training data above 0.1% of sampled entries. However, the pre-trained HA still outperforms train test FB results as it has more data to learn from multiple random graphs in the former case at higher missing percentages.

Note, that the *No training* results are better than the *Trivial replace with 1* results because even though the autoencoder is not trained, the *dimension* of the hidden layer encodes information about the graph itself. In particular, the choice of the hidden layer dimension is based upon experience with such graphs. The surprising result is that this single number (the dimension of the hidden layer) appears to provide substantial information, even for real-world networks and randomly initialized weights.

7.5 Conclusion

Pre-training technique is used with Hadamard Autoencoder to predict missing node-pair distances in social networks. The imputation technique trains on networks with variation in parameters such as the number of edges and probability of triangle creation, so that the new test network is treated as just another variation of the training set. As real-world network measurements can be very sparse, we focus on the efficient recovery of and from ultra-sparse samples of network measurements. The training data is sparse as it contains only a small fraction (from 80% to as low as 1%, 0.5%, 0.1%, and 0.0001%) of random entries from the complete distance matrix. System performance has been evaluated on randomly generated new Barabási-Albert and Powerlaw cluster networks as well as on a real-world Facebook test network. The effectiveness of the imputations is assessed based on the mean and absolute hop-distance errors in the predicted distance matrix. We also compare our results with a state-of-the-art Low-rank Matrix Completion method which can recover missing network measurements using partial observations. Three Trivial case results are also used for cross verification of the system performance.

The proposed model of pre-trained Hadamard Autoencoder exceeds the performance of Low-rank Matrix Completion by a wide margin for ultra-sparsely sampled data, i.e., for 1% and even 0.1% of total measurements. The results show that our technique performs well even when the number of sampled entries is very small. The mean error at 1% sampled entries is about 20% for pre-trained HA model and around 60% for the LMC model when tested on the Facebook network. The low error values show that HA can capture and recover intrinsic network properties by learning from simulated networks which share similar network features as the test network. It also overcomes the limitation of insufficient network data due to inaccessibility or complexity. This allows us to infer ultra-sparsely sampled networks with high confidence. Our results bolster the generative properties of autoencoders.

Though the autoencoder does not perform as well as the LMC approach for higher sampling percentages, we argue that this might be improved with improved/ extended training dataset. This can be investigated further in the future. Even though we demonstrate results using only two types

of networks, the use of HA can be extended to other types of networks for predicting missing node-pair distances without requiring any observations from the actual data. In the next chapter, fine-tuning of HAs is discussed to improve imputation accuracy. HAs may also be trained to make imputations on real-world directed networks, in the future.

Chapter 8

Building a Pre-training Oracle for Predicting Distances in Social Networks

In this chapter, we propose a novel method to make distance predictions in real-world social networks. As predicting missing node-pair distances is a difficult problem, we take a two-stage approach. Structural parameters for families of synthetic networks are first estimated from a small set of measurements of a real-world network and these synthetic networks are then used to pre-train the predictive neural networks. Since our model first searches for the most suitable synthetic graph parameters which can be used as an "oracle" to create arbitrarily large training data sets, we call our approach "Oracle Search Pre-training" (OSP).

Many real-world networks exhibit a Power law structure in their node degree distribution, so a Power law model, for example, can provide a foundation for the desired oracle to generate synthetic pre-training networks if the appropriate Power law graph parameters can be estimated. Accordingly, we conduct experiments on real-world Facebook, Email, and Train Bombing networks and show that OSP outperforms models without pre-training, models pre-trained with inaccurate parameters, and other distance prediction schemes such as Low-rank Matrix Completion. *In particular, we achieve a prediction error of less than one hop with only 1% of sampled distances from the social network.* OSP can be easily extended to other domains such as random networks by choosing an appropriate model to generate synthetic training data, and therefore promises to impact many different network learning problems.

8.1 Introduction

The shortest node-pair distances in graphs pack a lot of insight regarding the network as a whole and thus have been used in several graph analysis algorithms. In social network analysis, for example, hop-distances are used for predicting friendships [60, 61], extracting sensor network

topology [82, 87], and modeling social networks [216, 217] to name a few. However, missing entries in sampled datasets adversely affect the structural properties of social networks [20,21] and the original network characteristics are misrepresented. Predicting these missing entries would solve this problem.

Neural networks have proved to be useful in solving many prediction problems but they often require a large amount of training data to perform well. Pre-training has alleviated this issue with transfer learning, especially when the second task is much more difficult for training [74, 75]. However, relatively little is known about the pre-training behavior related to neural networks used for social network analysis as social networks are difficult to measure or sample completely due to their cost of measurement, privacy, or storage issues. The ever-increasing size of social networks today only adds to this problem.

We demonstrate a prediction model based on efficient pre-training for distance inference in social networks when only sparse samples are known. We use artificially simulated training data to compensate for the lack of sufficient real-world measurements. For accurate prediction, this artificial pre-training data have to be faithful to the characteristics of the target social network. However, we do not know the characteristics of the target network. All we have is a small fraction of randomly measured node-pair distances. A way to estimate the right pre-training parameters is required. The central idea of our method is to build an "Oracle" that points us towards this suitable set of pre-training parameters.

Fig. 8.1 gives a preview of our results for a 1133 node real-world Email network when only 0.5% of the total node-pair distances are known. We can see that accurate pre-training parameters improve the prediction performance by 33% over pre-training with inaccurate parameters and by 53% over models without pre-training.

We list the main contributions of our work in Section 8.2. Section 8.3 discusses relevant literature. Required theoretical background is provided in Section 8.4 with the complete methodology in Section 8.5. Sections 8.6 and 8.7 respectively, provide performance analysis and sensitivity study of the model. We summarize our work in Section 8.9.

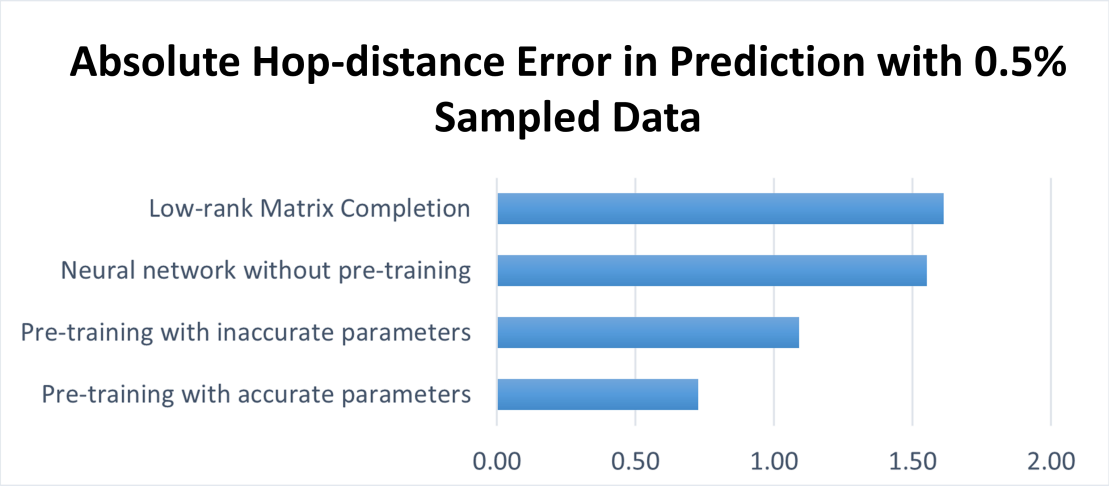


Figure 8.1: Comparison of proposed method with Low-rank Matrix Completion, neural network without pre-training, and pre-training with inaccurate parameters on a real-world Virgili Emails network when only 0.5% of distances are sampled. OSP improves the performance by 53% from a neural network without pre-training and by 33% from when inaccurate pre-training parameters are used.

8.2 Contribution

The main objective of this work is to present a sophisticated method of efficiently pre-training an autoencoder (AE) for prediction when sufficient target measurements are not available for training. A multi-stage setup for estimating network characteristics from partial distance measurements is proposed. We integrate the idea of pre-training with that of the best parameter selection for optimal prediction.

These optimally pre-trained autoencoders are demonstrated to predict missing shortest node-pair distances in social networks. We show the effectiveness of OSP on real-world social networks such as Virgili Emails, Facebook, and Train Bombing network which vary in their size and areas of function. The prediction performance is also compared against an existing distance inference model based on Low-rank Matrix Completion [87, 122, 218] (Section 8.6).

We improve with respect to the existing work in the following ways:

1. Artificially generated training data is used to pre-train an autoencoder when sufficient real-world data is not available (Section 8.5.1).

2. A novel two stage oracle is presented to predict suitable parameters for optimal pre-training. The oracle points us towards the suitable parameters solely using the sampled distances and without knowing any characteristics of the target network (Section 8.5.5).

3. OSP does not require large amounts of measurements from the target network that needs to be predicted. Our experiments show results for networks sampled as low as 0.1%, 0.5%, and 1% while also covering up to 80% sampling (Section 8.6).

4. The sensitivity of the predictive model is studied towards different pre-training parameters. This helps us understand pre-training behavior of social networks and identify the best set of pre-training parameters for the given test network (Section 8.7).

8.3 Related Work

In this Section, we discuss the most relevant literature to our work: prediction techniques for node-pair distances, completing missing network measurements from partial measurements, and using pre-training for predicting distances.

Low-rank Matrix Completion (LMC) [122] has been used effectively in reconstructing missing node-pair distances in graphs such as IoT or social networks when partial distance measurements are known, either randomly or from a selected set of landmarks. It has demonstrated accurate prediction up to 20-40% of sampled measurements [44, 87, 218]. This deterministic technique works well due to the low-rankness of the distance matrices. Thus we use Low-rank Matrix Completion as one of the reference points for comparing our results.

Deep learning techniques have found their way into social network analysis for distance prediction. Graph convolutional neural networks have been used to predict missing links in recommender systems [72]. Preserved distances of each node from a few landmark nodes are successfully used [219] to train a multi-layer perceptron (MLP) and predict the distance between two vertices without a high space cost. WiDE, [220], uses unsupervised stacked autoencoders to pre-train a deep neural network and predicts person-to-person distances using surrounding Wifi signals. However, these techniques rely on a significant amount of measurements within the real-world networks.

Our work is inspired by the methods proposed in [221, 222] where Hadamard Autoencoders are trained on synthetic data when only sparse measurements are available from the target network. Though this helps us train on artificial data, all social networks are not the same and vary greatly in their characteristics. This leaves room for improving the pre-training process to be customized for each social network being reconstructed.

Thus, we hypothesize that building a pre-training "Oracle" that guides us into a narrower range of the pre-training data being used will help significantly improve the prediction performance by generating customized training data that is more faithful to the given social network.

8.4 Theory

In this Section, we provide necessary background by giving a brief introduction of the autoencoders and the data used. We also briefly talk about the concept of pre-training which forms the foundation of this work.

8.4.1 Supervised Autoencoders

An autoencoder is a special type of artificial neural network that tries to reproduce the input at the output while learning its representation in a low dimensional space. It consists of two parts, namely, encoder and decoder. The encoder takes the input, **x**, and maps it to a hidden representation **h**, such that:

$$\mathbf{h} = \sigma(\mathbf{W}\mathbf{x} + \mathbf{b}), \tag{8.1}$$

where **W** is the weight matrix, **b** is the bias vector and σ is the activation function for the encoder. We use Leaky ReLU [223] as the activation function throughout the autoencoder. It is defined as:

$$\sigma(z) = \begin{cases} z, & \textit{if } z > 0 \\ \alpha z, & z <= 0, \end{cases} \tag{8.2}$$

where z is the input and α is a small non-zero, constant gradient. Leaky ReLU helps to solve the

"vanishing gradient" problem caused by ReLU [224] where the function outputs 0 if $z < 0$, seizing the update of the hyperparameters and thus the learning.

The decoder reconstructs this hidden representation $\mathbf{h}$ to the output, $\hat{\mathbf{x}}$, given by:

$$\hat{\mathbf{x}} = \sigma(\mathbf{W}'\mathbf{h} + \mathbf{b}'), \tag{8.3}$$

where $\mathbf{W}'$ is the weight matrix, $\mathbf{b}'$ is the bias vector and σ is the activation function for the decoder.

The input and output layers have the same number of neurons in order to reconstruct the given input. Each neuron operates on the incoming signal and generates an output based on its activation function, weight, and bias values. The weight and bias values are found by training so that the difference between $\hat{\mathbf{x}}$ and $\mathbf{x}$ is minimized.

8.4.2 Training Protocols

We use the supervised training method to train the hyperparameters, i.e., while training, the expected output is known. Back-propagation [225] is used to train the weights and biases for each layer during training. Back-propagation consists of two phases. During the first phase, input is fed to the autoencoder with weights and biases initialized to random values and the output layer reconstructs the input. In the second phase, the error between expected and predicted output is calculated using a mean squared error loss function given by Eq. (8.4). The weights and biases are updated iteratively using this error value.

$$\mathcal{L}(\mathbf{x}, \hat{\mathbf{x}}) = ||\mathbf{x} - \hat{\mathbf{x}}||^2, \tag{8.4}$$

where $\mathbf{x}$ is the known expected output, $\hat{\mathbf{x}}$ is the predicted output as reconstructed by the output layer. Thus from Eq. (8.1), (8.3), and (8.4), we can say that

$$\mathcal{L}(\mathbf{x}, \hat{\mathbf{x}}) = ||\mathbf{x} - \sigma(\mathbf{W}'(\sigma(\mathbf{W}\mathbf{x} + \mathbf{b})) + \mathbf{b}')||^2. \tag{8.5}$$

In other words, the loss function is the mean $(\frac{1}{n}\sum_{i=1}^{n})$ of the squared errors $(x_i - \hat{x}_i)^2$, which is an easily computable quantity for the given data sample, and can be expressed as $MSE = \frac{1}{n}\sum_{i=1}^{n}(x_i - \hat{x}_i)^2$, where n is the total number of samples.

The size of the hidden layer can be adjusted in proportion to the size of the target data. We used learning rate 0.001, batch size 1, and hidden layer size 50 for the Facebook network and 20 for the Virgili Emails and Train Bombing network, as determined from the observed measurements.

8.4.3 Pre-training

The concept of pre-training is inspired by the way humans learn. While learning new things, we re-use our knowledge of the past. Pre-training a neural network refers to training the model during one task and using the trained parameters as a head start for another task. This way, the model does not have to learn from scratch for the second task. It is especially useful if the two tasks are similar or the final task is much more difficult and can be broken down into simpler pre-training tasks [74,75]. For instance, training an autoencoder to predict missing entries of sparsely available data can be difficult. Instead, we pre-train the autoencoder on readily available simulated data and try to predict the parameters for this pre-training data so that it is similar in characteristics to the target data.

Analysis of both natural and human-created real-world networks have shown that such networks closely follow specific rules, such as Power law behavior in their degree distributions $P(k) \sim k^{-\xi}$, with the exponent varying between 2 and 3. The characteristic that only a few nodes have an extremely large number of connections while most of the nodes have very few links is due to 'preferential attachment' [208,209,211]. Furthermore, social networks like Facebook and WeChat are observed to have Power law growth dynamics [210]. Thus, for a given real-world network, it is possible to identify a closely related network type that can be used to generate synthetic data for training. Social networks that we are interested in for this work, e.g., Emails, Facebook, and Train Bombing networks, are well documented to follow Power law.

Training the autoencoder on Power law networks with similar parameters helps the model treat the target real-world network as just a variation of the training network. Another advantage is that we can generate sufficient data as required for proper training and effective performance of the neural network.

8.4.4 Data

Given a graph $G = \{V, E\}$, where V is the set of nodes in the network and E is the set of undirected edges, we use shortest hop-distances between node-pairs as our data. For a given graph G, its *shortest hop-distance matrix* $\mathbf{H}$ can be given as

$$\mathbf{H} = [h_{ji} = h_{ij} = \textit{shortest hop-distance from } i \textit{ to } j], \tag{8.6}$$

The shortest *Hop-Distance* Matrix (HDM), $\mathbf{H} \in \mathbb{N}_0^{N \times N}$ can represent the entire network, where $\mathbb{N}_0$ denotes the non-negative integers $\mathbb{N} \cup \{0\}$ and N denotes number of nodes in G.

Note that nodes and links may have a different interpretation in different networks, e.g., in the Facebook network, nodes are users and links represent friendships, whereas in a software system network, nodes represent software modules and links may represent dependency between the two modules. While our work focuses only on *connected, unweighted, and undirected graphs*, there are several commercially important social networks in this domain such as Friendship, Email, crime, and interaction networks to name a few.

8.4.5 Sampling Scheme

We aim to predict missing node-pair distances in networks when the target network is only partially sampled. In our experiments, we measure a small set of distances between random pairs of nodes from the distance matrix $\mathbf{H}$ to achieve distributed, unbiased sampling. The randomly sampled $\mathbf{H}$ results in an $N \times N$ matrix $\mathbf{P}$, as shown in Fig. 8.2, where the sampled entries are depicted by solid squares and '?'s represent missing entries. Other sampling schemes such as random walk or snow-ball sampling tend to be biased towards higher degree nodes, producing

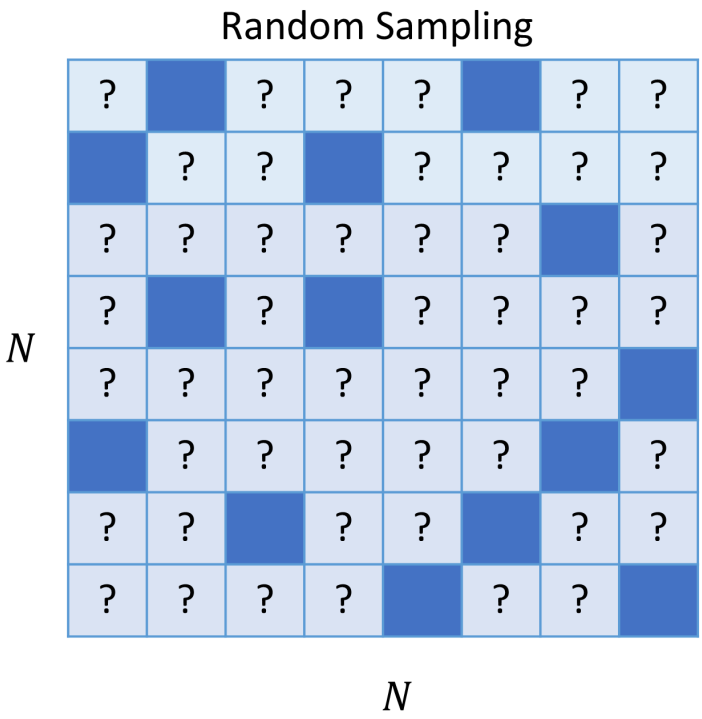


Figure 8.2: Visual illustrations of **P** (sampled **H**) for random sampling scheme with $N = 8$. Known entries are denoted by dark blue and unknown values by light blue.

a misrepresentation of typical large-scale, real-world, unstructured networks with non-uniform degree distributions, e.g., Power law degree distribution [198].

The conventional approach for network reconstruction from a sparse set of random samples is based on Low-rank Matrix Completion which requires a sampling scheme to measure at least one non-zero distance value in each row for the test network [87]. However, our approach overcomes this restriction.

8.4.6 Low-rankness of Data

Low-rankness in a dataset implies redundancy which allows reconstruction of missing node-pair distances from a small set of entries, i.e., partial measurements. We leverage this while reconstructing distance matrices.

Here, we demonstrate low-rankness of the real-world social networks used. Fig. 8.3 shows the logarithm of all the singular values of **H**. Only a small set of singular values are dominant with the

later magnitudes falling down rapidly towards '0', indicating low-rankness of the **H** matrices for shown networks.

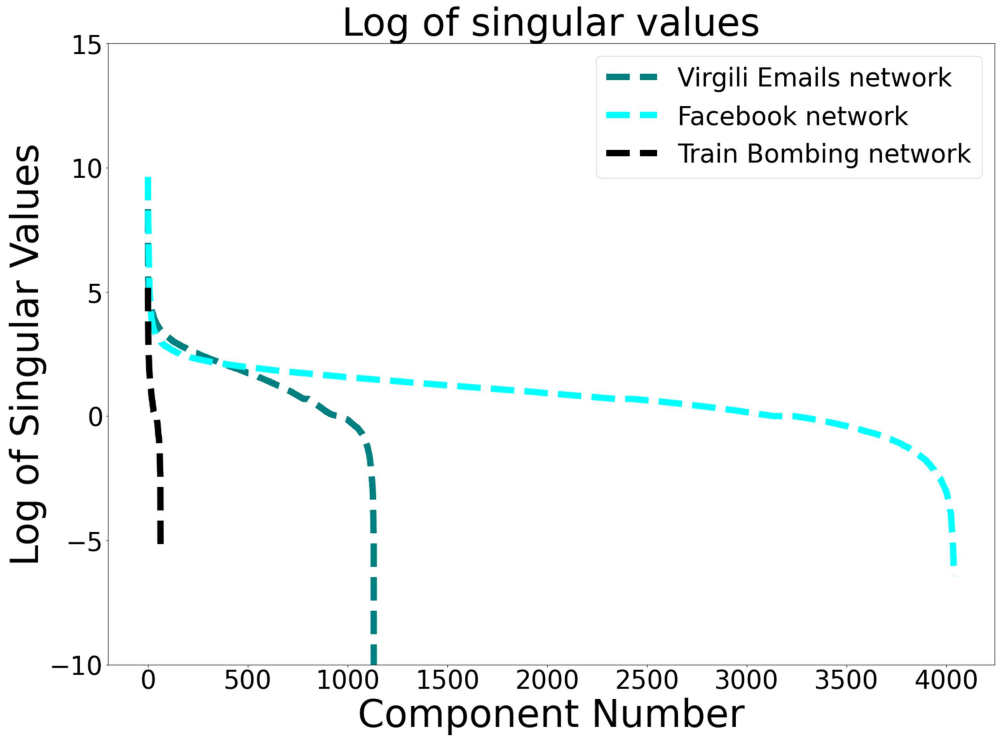


Figure 8.3: Low-rankness of hop-distance matrices of social networks used in this research.

Note that low-rank data lies on low-dimensional linear subspace whereas low-dimensional data lies on low-dimensional non-linear subspace.

8.5 Methodology

An overview of the proposed simple approach for predicting distances from a small set of measurements is shown in Fig. 8.5. In this Section, we discuss how pre-training data is artificially generated to train a neural network, which parameters are tuned to make the synthetic data similar to real-world data we are predicting, and how the ideal values of this parameter are estimated using our proposed pre-training "Oracle".

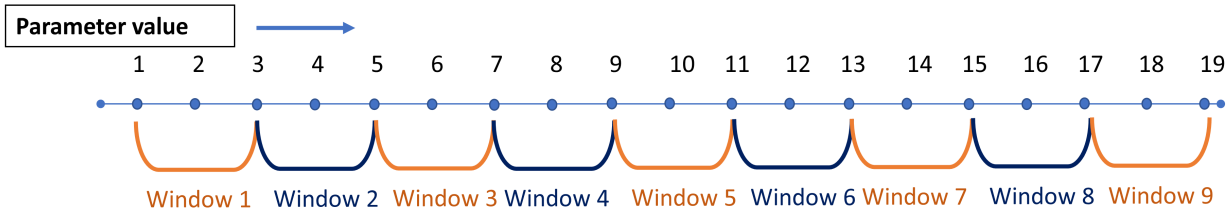


Figure 8.4: Illustration of selecting windows for parameter values for generating artificial Power law networks. Neural network is trained and tested to evaluate the effectiveness of each window on prediction.

8.5.1 Generation of Artificial Training Data

We learnt in Section 8.4.3 that our target networks, i.e., social networks, are Power law in nature. Thus, we use synthetic Power law networks as a base for generating our pre-training data.

Power law cluster model, in Python's ***Networkx*** library, creates graphs for given network size and edges using a preferential attachment with the function ***powerlaw_cluster_graph****(N, m, p)*, where a graph of N nodes is grown by attaching new nodes, each with m edges that are preferentially attached to existing nodes with high degree [212, 214], and p is the probability of forming a triangle after adding a random edge. The model generates a network whose degree distribution follows a Power law curve and thus, represents social networks such as protein-protein interactions, World-Wide-Web, citations network, Facebook, and others. However, depending on the given parameter values, the generated graphs could be vastly different from the specific network under consideration. Even for two networks of the same type, e.g., social networks, the parameters could be significantly different. Next, we address the method for determining the appropriate parameters that need to be tuned, corresponding to the specific network of interest.

8.5.2 Tuning Parameter for Synthetic Data Generation

Node degree is the simplest way of describing how nodes are connected. Yet, it has a prodigious effect on the network characteristics such as clustering coefficient, betweenness centrality, network diameter, etc., proving to be a simple but significant network feature.

As parameter m represents the number of edges connected to each new node, in the function ***powerlaw_cluster_graph****(N, m, p)*, it is analogous to the degree of the node. Hence, we focus

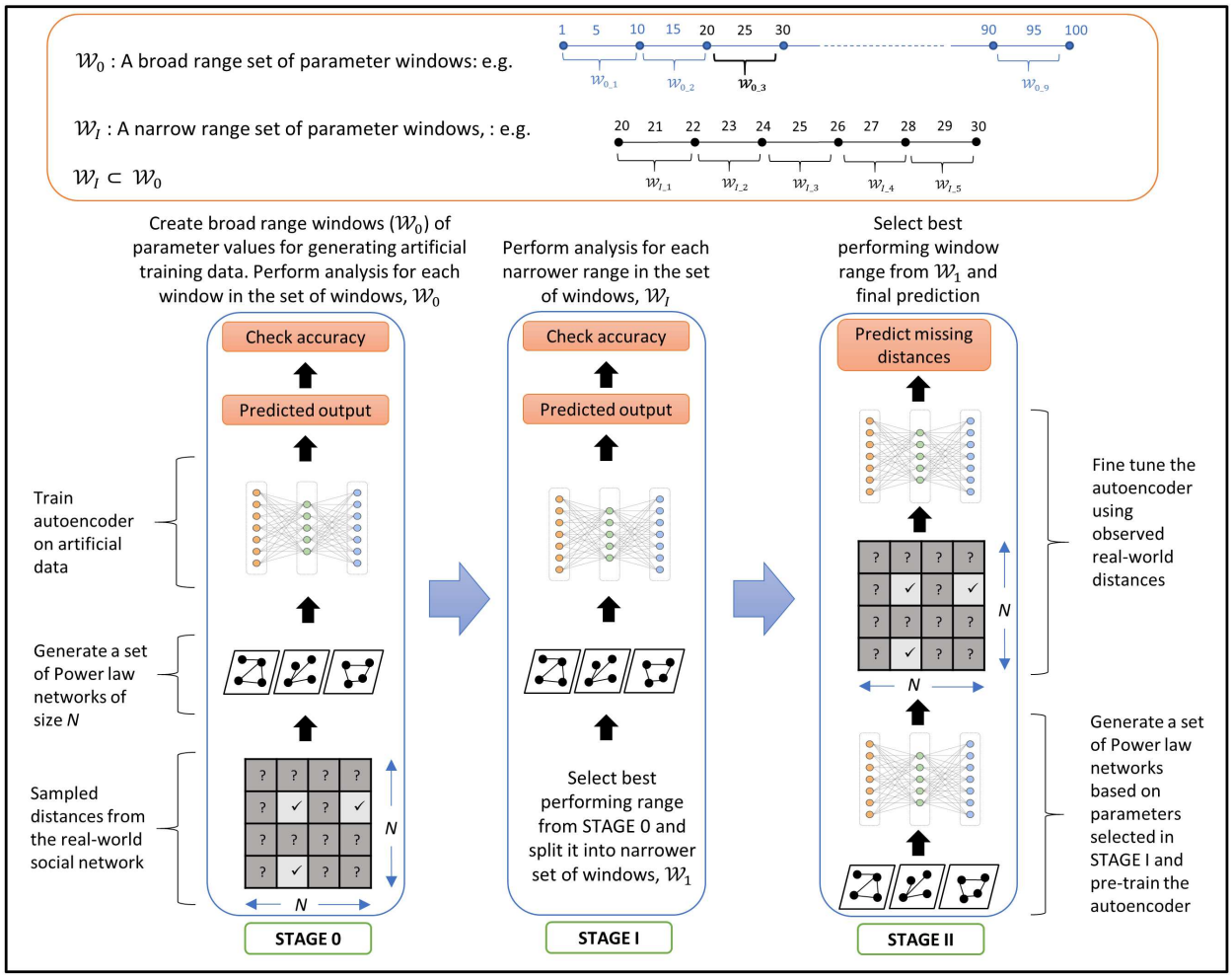


Figure 8.5: Oracle Search Pre-training (OSP): Model architecture. STAGE 0 helps narrow down suitable parameter values from a very broad range and STAGE I helps to select the three most suitable parameters for the given sampled network. The neural network is pre-trained on artificial data generated using these selected parameters and fine tuned using sampled distances form the target network in STAGE II to predict missing node-pair distances in the real-world social network.

on finding a suitable range of values for parameter m for the given target network. We generate various training networks with fixed N - to generate networks of the same size as that of the target network, consciously selected values of m - to study the effect of different m values on prediction performance - while using widely spread values of probability p, i.e., from 0.1 to 0.9 - to generate networks with different extent of clustering. This allows us to train the autoencoder on different clustering densities, as we do not know the clustering coefficient of our target network.

Next, we use the window method, as described below, to systematically vary m and study its effect on prediction performance.

8.5.3 Window Method for Varying the Parameter Value

Our goal is to find a narrow range of values for m to generate training data so that it comes closer to the characteristics of our target network. If we assume that the average node degree of the target network is less than some value n_d and that the network has a Power law node degree distribution, then the distribution is highly right-skewed and we can safely say that the set of suitable m values, for training data generation, are more likely to fall on the left side of average node degree as most of the nodes have node degree less than the average node degree of the network. We also demonstrate this in Section 8.7.

As $m > 0$ for connected networks, we partition the range 1 to n_d in small windows as shown in Fig. 8.4, where $n_d = 19$ for illustration purposes. An autoencoder is trained on networks generated for the m values in each window and is tested. This allows us to evaluate the effectiveness of each parameter window with respect to its prediction performance. The goal is to find an ideal range of parameters that will give us the best prediction for the given target network.

Note that the assumed value of n_d may vary as per the type of the target network. For example, we know that in a road network, a node (an intersection) is less likely to have more than 10 edges (roads) so n_d can be set to a lower value. **Setting a higher n_d has no major drawbacks in the model as it only produces a few more windows for analysis but underestimating n_d might make us miss the ideal set of values of m for the target data.**

8.5.4 Evaluation Parameters

We now describe the performance metric used to evaluate the prediction performance of our model. We aim to calculate the cumulative error in prediction. Thus, we measure and compare mean error and absolute hop-distance error to evaluate the prediction performance and to demonstrate the effectiveness of our model. Note that we measure these error values for the set of node-

pairs under test. This can be a set of observed node-pairs or unobserved node pairs depending on the stage of the model and will be mentioned accordingly.

8.5.4.1 Mean error

We define mean error M_e as follows:

$$M_e = \left[\sum_{\forall\ i,j \in \check{\Omega}} |\check{h_{ij}}(f) - h_{ij}| \right] \times 100 \Big/ \left[\sum_{\forall\ i,j \in \check{\Omega}} h_{ij} \right], \tag{8.7}$$

where, $\check{\Omega}$ is a set of all node pairs under test, $\check{h_{ij}}(f)$ refers to the predicted shortest hop-distance from node i to node j when f percentage of random measurements are sampled and h_{ij} refers to its original expected value. Mean error gives a percentage value of the prediction error with respect to sum of the original hop-distances in the network.

8.5.4.2 Absolute hop-distance error

Absolute hop-distance error (AHDE) is defined as

$$H_e = \left\{ \sum_{\forall\ i,j \in \check{\Omega}} |\check{h_{ij}}(f) - h_{ij}| \right\} \Big/ \{card(\check{h})\}, \tag{8.8}$$

where, $card(\check{h})$ gives the total number of node-pair entries in **H** under test. AHDE captures average deviation in prediction in the magnitude of hop-distances. For instance, the absolute hop-distance error of 1 implies that on average a predicted path length will be off by 1 hop, i.e., for the original hop-distance of 10, the predicted value will be somewhere from 9 to 11.

8.5.5 Model

A visual illustration of the model architecture can be seen in Fig. 8.5. A detailed explanation of various stages of the proposed model follows where we explain the ideal parameter selection (STAGE 0 and STAGE I), systematic pre-training and final prediction of missing node-pair distances (STAGE II).

8.5.5.1 STAGE 0-Broad range parameter selection

In the initial stages of building the Oracle, the autoencoder is evaluated for a range of m values as we are not aware of the ideal values for the tunable parameter. These values are chosen using a sliding window to include a wide range (Section 8.5.3 and Fig. 8.4). As we saw in Section 8.5.3, a parameter range can be assumed for m to start. If the range is broad, it can be divided into smaller windows. This can be viewed as STAGE 0.

For example, being a communication network, the Virgili Emails network might have a high average node degree. So we first test the autoencoder performance on a set of windows $\mathcal{W}_0$ of m values of [1,5,10], [10,15,20], and so on up to [90,95,100] (as shown in Figs. 8.6[a,b]). The performance is evaluated of both mean error and absolute hop-distance error. The best performing window of [1,5,10], say $\mathcal{W}_{0\_k}$, is selected from this set for the next stage, i.e., STAGE I.

Note that STAGE 0 is optional and can be used when a quick narrow down of a possible wide range of m values is required.

8.5.5.2 STAGE I-Narrow range parameter selection

The best performing window from STAGE 0 is further split into a narrower set of windows, $\mathcal{W}_I$ and $\mathcal{W}_I \subset \mathcal{W}_0$ (as shown in Fig. 8.5), to tune the parameter values further. The autoencoder is trained for each of these windows, $\mathcal{W}_{I\_k}$, independently and is tested after each session on observed distances from the target network.

For example, we selected the range [1,5,10] from STAGE 0 for the Virgili Emails network, thus, in STAGE I, we have trained the autoencoder on narrower windows of m values as shown in Fig. 8.7[a,b] from [1,2,3], [3,4,5], and so on up to [9,10,11].

As we are evaluating the system over partial and presumably a very small quantity of sampled data, and because no single window performed significantly better than the rest in STAGE I, we select the top three performing parameter windows for the sampled measurements. You can see in Figs. 8.8 and 8.11 that windows [5,6,7], [7,8,9], and [9,10,11] perform the best for the Virgili Emails and Facebook network. The Train Bombing network being small in size, we can safely pick a narrower range and train the final model on the best performing window from STAGE 1.

8.5.5.3 STAGE II-Final Prediction

The best parameter values selected from STAGE I are eventually used to generate pre-training data. Once trained, this model can be fine tuned on observed distances from the target network. We examine the results in both cases (with and without fine tuning) in Section 8.6. The trained autoencoder then reconstructs the complete distance matrix from only a fraction of sampled distances from the test network.

Note that STAGE 0 and STAGE I act as an "Oracle" to search the ideal pre-training parameters corresponding to the target network while STAGE II pre-trains AE on artificial data generated with these parameters. STAGE II results are shown in Fig. 8.8 for Virgili Emails, Fig. 8.11 for Facebook, and Fig. 8.14 for the Train Bombing network, and discussed in Section 8.6.

8.5.6 Implementation Details

We used TensorFlow, Google's Machine Learning Library, to build the autoencoder and wrote the script in Python 3.6. We ran the experiments on Google Colab.

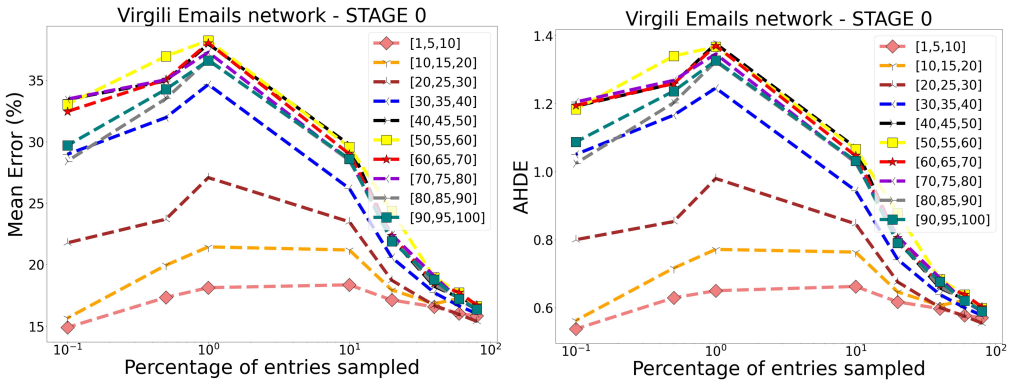
Data used and the code for our model are available on Github: https://github.com/gunjanmahindre/Pre-training-Oracle

8.6 Performance evaluation of the proposed method

We measure mean error (Section 8.5.4.1) and absolute hop-distance error (Section 8.5.4.2) to evaluate the prediction performance of our model and compare it with the Low-rank Matrix Completion [87] based approach. Here, the errors are measured only over the unobserved node-pair distances. We know that all diagonal values in a distance matrix are '0' and no off-diagonal entries are '0', so we set all diagonal values of $\hat{\mathbf{P}}$ to 0 and round-up all off-diagonal values between 0 and 1 to 1.

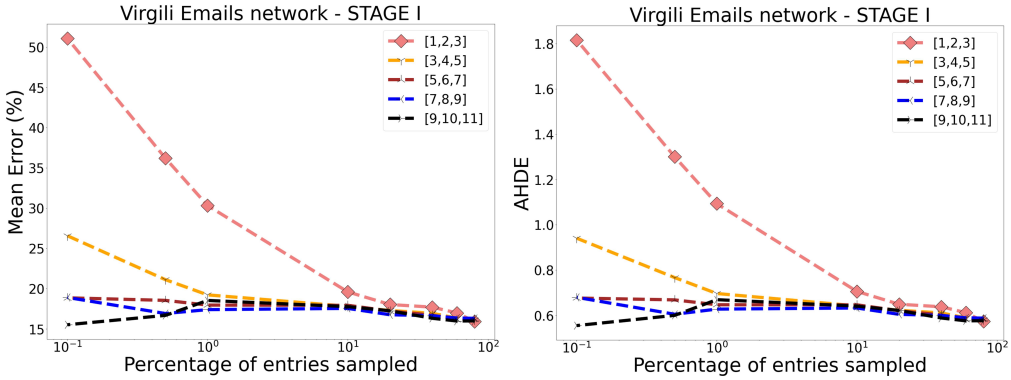
The model has been evaluated for different variations and are labeled as shown in Figs. 8.8, 8.11, and 8.14. These labels are explained below:

- **Trivial 0**: all missing values are replaced by 0



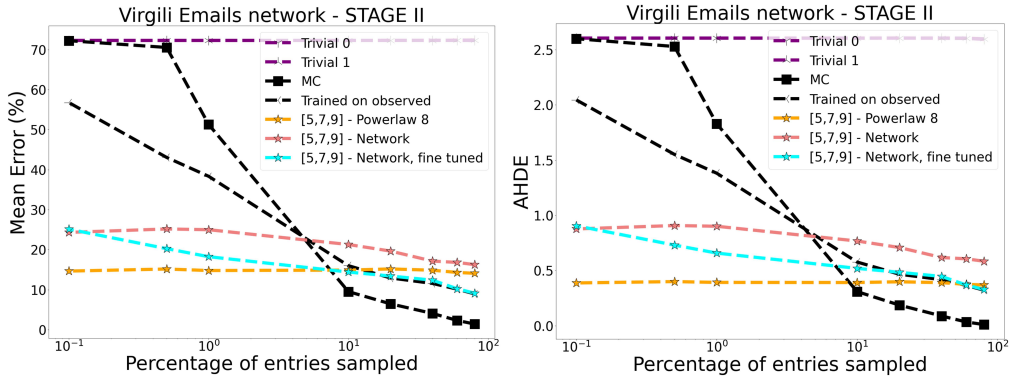
(a) **(b)**

Figure 8.6: Parameter selection for pre-training Oracle - Virgili Emails network: Each broad range window is evaluated for its prediction efficiency on sampled distances. The best parameter range is selected for the next stage, hence, parameter range [1,5,10] is selected. (a) STAGE 0: Mean error (b) STAGE 0: AHDE.



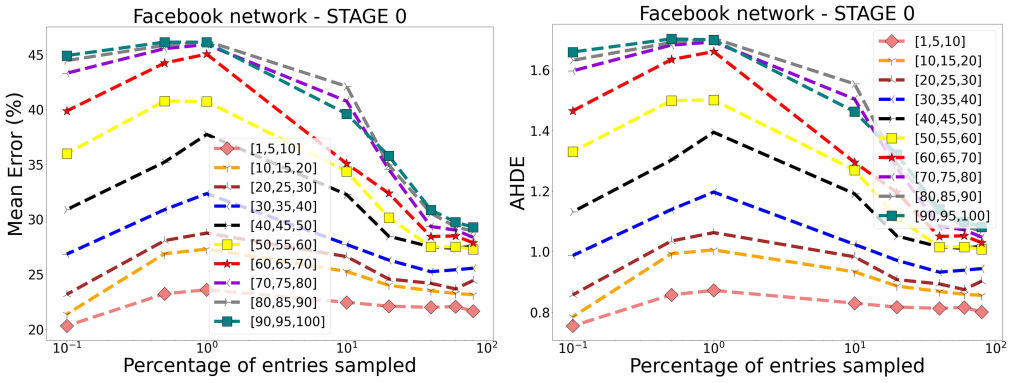
(a) **(b)**

Figure 8.7: Parameter selection for pre-training Oracle - Virgili Emails network: The braod range of (1,5,10) selected from STAGE 0 is split into narrower windows (a) STAGE I: Mean error (b) STAGE I: AHDE.



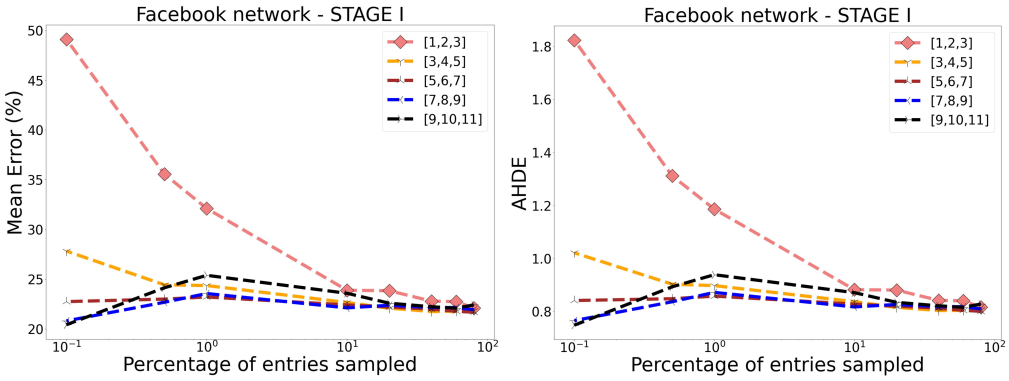
(a) **(b)**

Figure 8.8: Virgili Emails network: Prediction performance. Pre-trained neural networks (yellow, pink, and blue) outperform prediction especially at lower sampling percentages from the real-world network while achieving graceful degradation. Fine tuning along with pre-training gives higher accuracy than a neural network trained only on sampled entries.



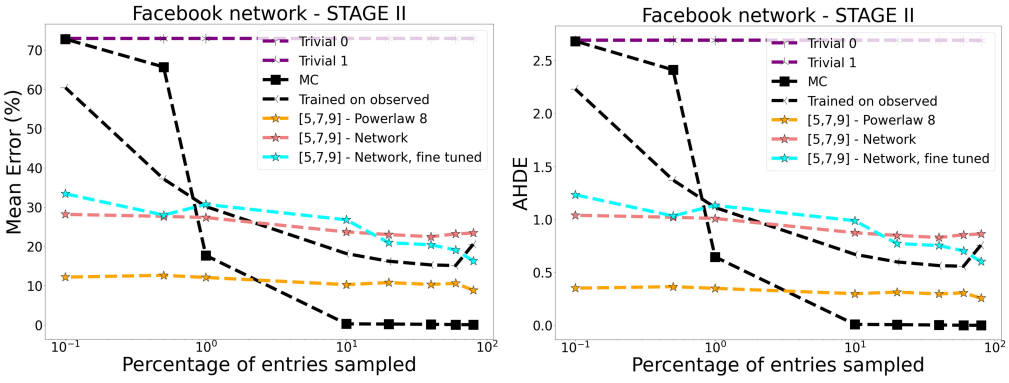
(a) **(b)**

Figure 8.9: Parameter selection for pre-training Oracle - Facebook network: Each broad range window is evaluated for its prediction efficiency on sampled distances. The best parameter range is selected for the next stage, hence, parameter range [1,5,10] is selected. (a) STAGE 0: Mean error (b) STAGE 0: AHDEr.



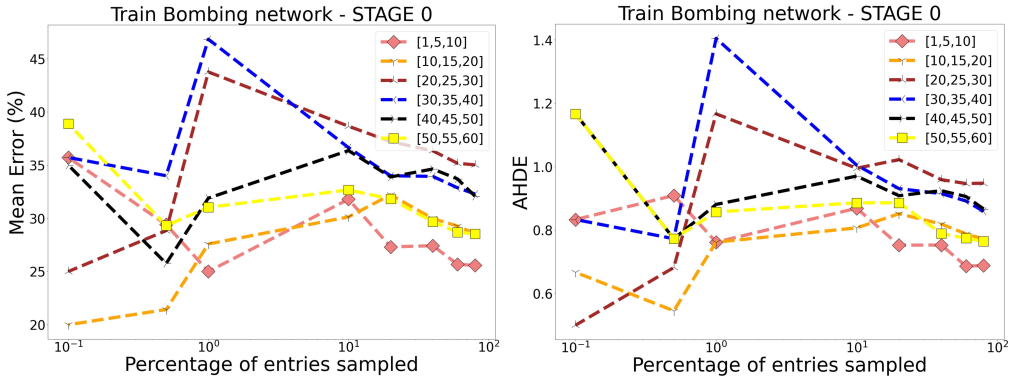
(a) **(b)**

Figure 8.10: Parameter selection for pre-training Oracle - Facebook network: The braod range of (1,5,10) selected from STAGE 0 is split into narrower windows (a) STAGE I: Mean error (b) STAGE I: AHDE.



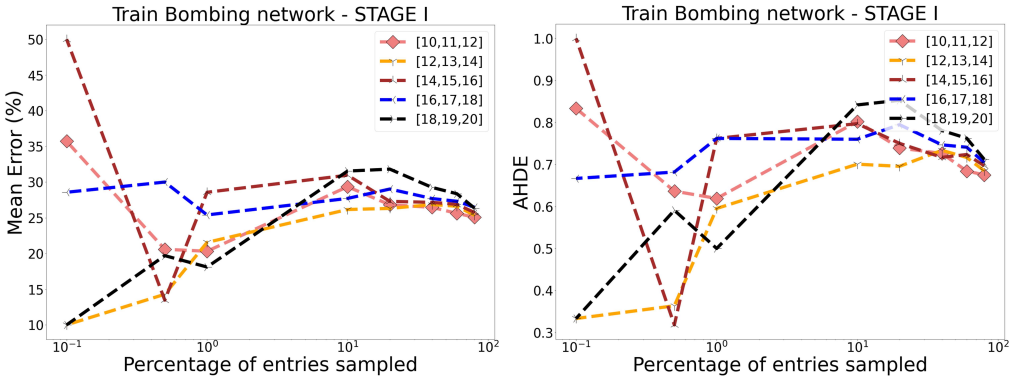
(a) **(b)**

Figure 8.11: Facebook network: Prediction performance. Pre-trained neural networks (yellow, pink, and blue) outperform prediction especially at lower sampling percentages from the real-world network while achieving graceful degradation. Fine tuning along with pre-training gives higher accuracy than a neural network trained only on sampled entries.



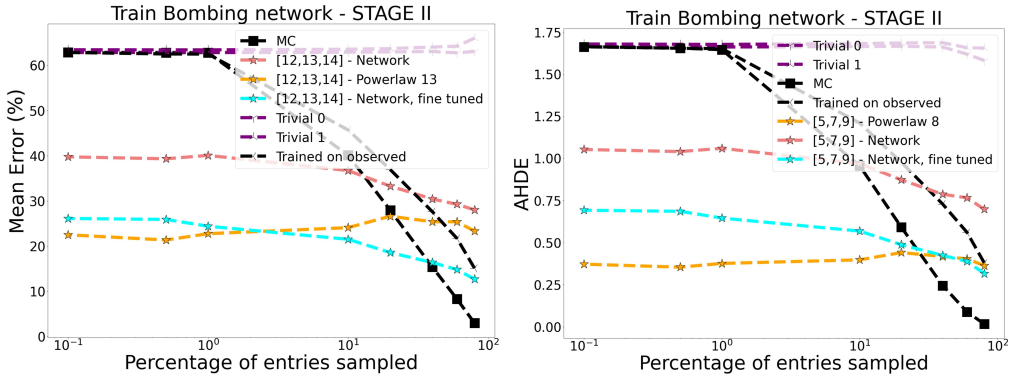
(a) (b)

Figure 8.12: Parameter selection for pre-training Oracle - Train Bombing network: The best performing parameter range, [1,5,10], is selected for the next stage (a) STAGE 0: Mean error (b) STAGE 0: AHDE.



(a) (b)

Figure 8.13: Parameter selection for pre-training Oracle - Train Bombing network: The braod range of (1,5,10) selected from STAGE 0 is split into narrower windows (a) STAGE I: Mean error (b) STAGE I: AHDE.



(a) (b)

Figure 8.14: Train Bombing network: Prediction performance. Pre-trained neural networks (yellow, pink, and blue) outperform prediction especially at lower sampling percentages while achieving graceful degradation. Fine tuning along with pre-training gives higher accuracy than a neural network trained only on sampled entries.

- **Trivial 1**: all missing values are replaced by 1

  Trivial 0 and Trivial 1 cases provide a reference to the worse case model performance.

- **MC**: prediction using Low-rank Matrix Completion

- **Trained on observed**: the AE has been trained only on the sampled node-pair distances from the real-world network and no pre-training has been done at all

- **[x,y,z] - Powerlaw q**: the STAGE II AE, i.e., trained on artificial Power law networks with parameter m values x,y, and z, was tested on a same size Power law network artificially generated with $m = q$

- **[x,y,z] - Network**: the STAGE II AE, i.e., trained on artificial Power law networks with parameter m values x,y, and z, was tested on the real-world social network

- **[x,y,z] - Network, fine tuned**: the STAGE II AE, i.e., trained on artificial Power law networks with parameter m values x,y, and z, was fine tuned with the observed entries of the real-world network and then tested on the unobserved entries of the same real-world network

Now let us take a look at the performance of the model for three real-world social networks.

Virgili Emails network: This is a 1133 nodes network with average node degree of 9.6.

Facebook network: This is a 4039 node network with average node degree of 43.6.

Train Bombing network: This is a 64 node network with average node degree of 7.5.

From STAGE 0 and STAGE I of the Virgili Emails network, as shown in Fig. 8.6 and Fig. 8.7 respectively, three best windows were selected, [5,6,7], [7,8,9], and [9,10,11]. In STAGE II, the AE is thus trained on [5,7,9] to cover these three windows. This behavior is also seen in the Facebook network and thus a similar process is followed in STAGE 0 and STAGE I for the Facebook network. For the Train Bombing network, we were able to select the window of [12,13,14]. The STAGE 0 and STAGE I results for the Facebook network can be seen in Fig. 8.9 and Fig. 8.10 respectively.

Note that the parameter values selected by the "Oracle" are not the same as the average node degree of the test network but tend to be on the lower side of the average node degree

value. We will explore this relation and the sensitivity of the model towards parameter value m in Section 8.7.

The STAGE II (Virgili Emails: Fig. 8.8, Facebook: Fig. 8.11, Train Bombing: Fig. 8.14) show that AE performs far better than MC for lower percentages of sampled measurements. We also see that AE trained on artificial data performs better than AE trained only on the observed entries as the neural network has much more data to train on from synthetic Power law networks. Note that the network performs best for a variation of Power law network (see label "[x,y,z]-Powerlaw") which is obvious as the network was trained on very similar data. The error, when tested on the real-world network is slightly higher than for "[x,y,z]-Powerlaw" as the real-world network is different than the synthetic Power law networks. However, we notice that fine tuning the AE almost always bridges this gap. Especially when we have a higher percentage of sampled node-pair measurements from the network, fine tuning gives lower error than mere Power law trained AE for all three networks.

It is worthwhile to say that OSP performs well not only for large networks but also for smaller networks such as the Train Bombing network here, showing independence of the model performance with respect to the network size.

Bubble plots:

We use the bubble plot to further analyze our prediction performance. We plot the actual number of entries for the given magnitude on the X-axis and the number of entries for that magnitude in the predicted matrix on the Y-axis. The ideal plot should give all bubbles on the $y = x$ line indicating that the number of entries in the actual and the predicted matrix are the same for any magnitude of hop-distance. However, the real plots are obviously expected to be a deviation of this ideal scenario.

We analyze the prediction results of a real-world network, the Virgili Emails network, and an artificially generated network, a Power law network. We compare the prediction results of both autoencoder as well as the Low-rank Matrix Completion approach to understand how the

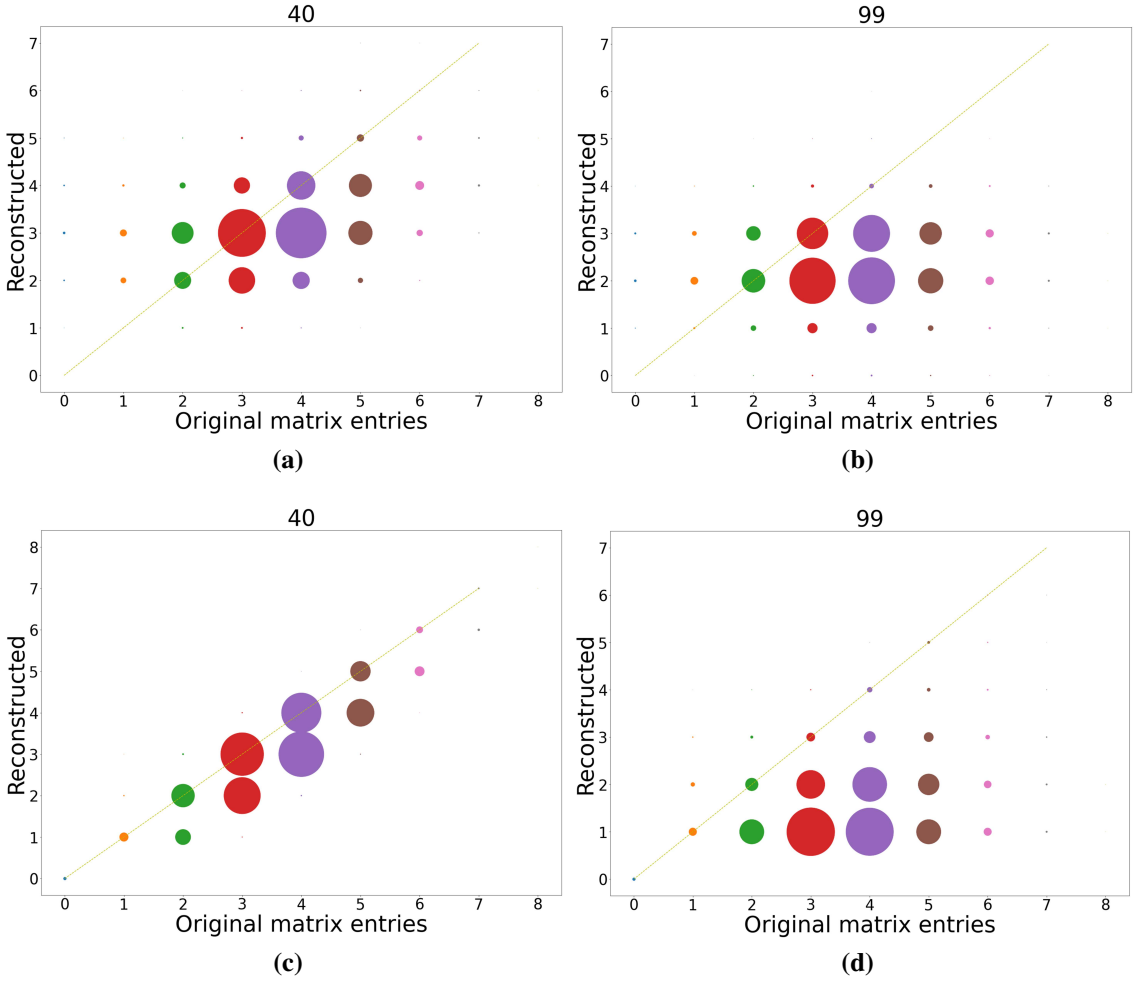


Figure 8.15: Bubble plot for Virgili Emails network: (a,b) comparison of Autoencoder trained on $m = [5,7,9]$ and (c,d) Matrix Completion for 40% and 99% missing entries respectively

performance differs at a micro level. Two scenarios are compared for each method, (a) when 40% of entries are missing from the distance matrix before prediction and (b) when 99% of entries are missing from the distance matrix before prediction. This is a broad enough spectrum to compare the performance of our deterministic (LMC) and non-deterministic (autoencoder) approaches for prediction.

We can see from Fig. 8.15 [a,b] and Fig. 8.16 [a,b] that the bubbles are larger on the axis or closer to the axis. This means that LMC performs better than autoencoder in maintaining the number of entries close to the actual when only 40% of entries are missing. This behavior is observed for both the Virgili Emails network and the Power law network.

However, as we move towards a higher percentage of missing entries, i.e., for 99% missing entries, this alters. We can see in Fig. 8.15 [c,d] and Fig. 8.16 [c,d] that autoencoder performs better than Low-rank Matrix Completion here. This can be seen as the larger bubbles are mainly at the lowest for LMC wherea,s for autoencoders, the largest bubbles are not at 1. This means that LMC predicts most of the entries as 1, 2, and 3, in that order, whereas autoencoder predicts most of the entries as 2,3,1 in that order. This order might change as per the network and actual count for those magnitudes, however, we can see that there is a clear difference in the way these two methods predict. Autoencoders can predict better than LMC for higher missing data percentages.

Note that while comparing the count of entries for a certain magnitude, we are losing track of the location of that entry. In other words, in the cases where the count is the same, there is no guarantee that the exact same entries are recovered to the exact same magnitude. If there is a high change in the actual magnitude of the entries but overall the number of entries of a given magnitude is the same, the plot will look like an ideal plot while the prediction results are nowhere near ideal. However, we do not use these plots as our primary basis of performance analysis but to further understand the micro effect of different methods at various percentages of sampling. These plots help us understand the distribution of distances before and after the prediction in a different light.

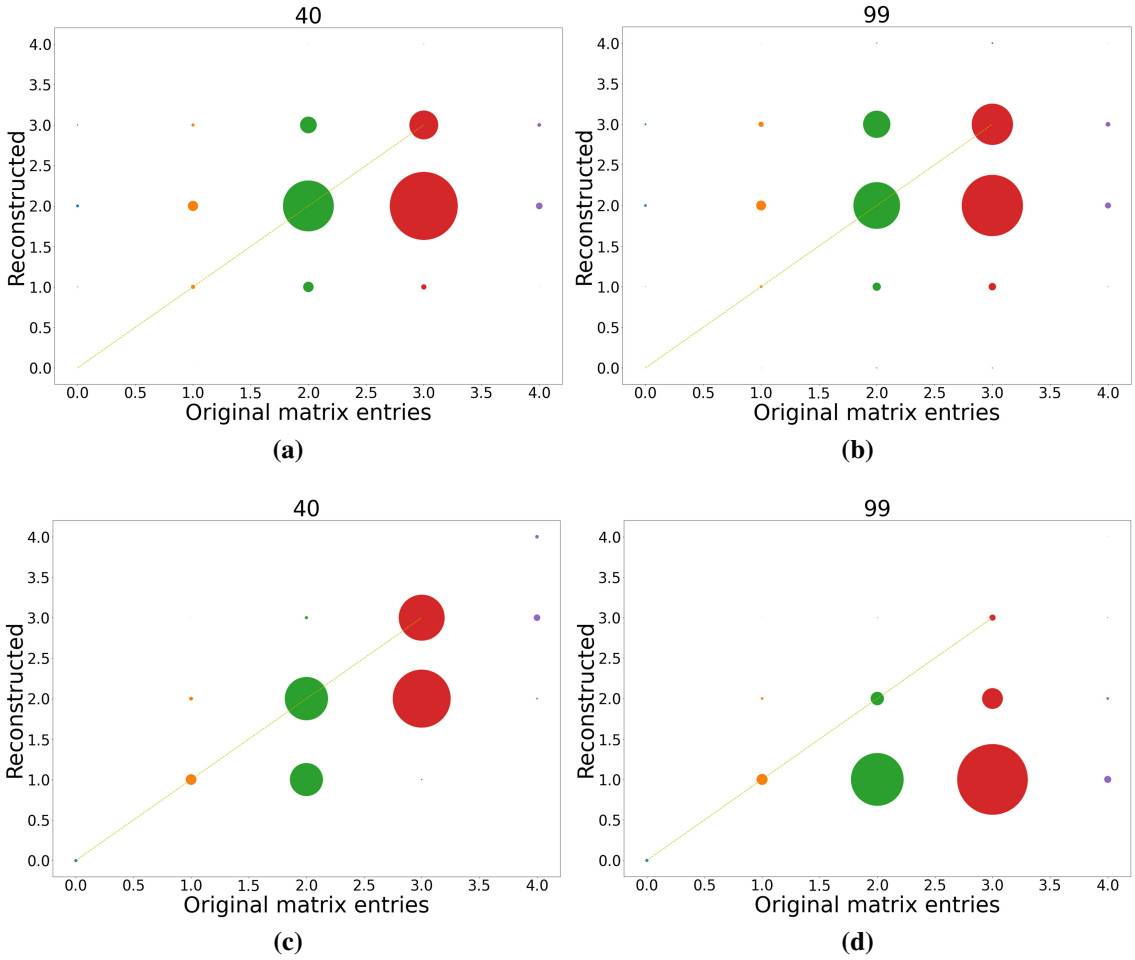
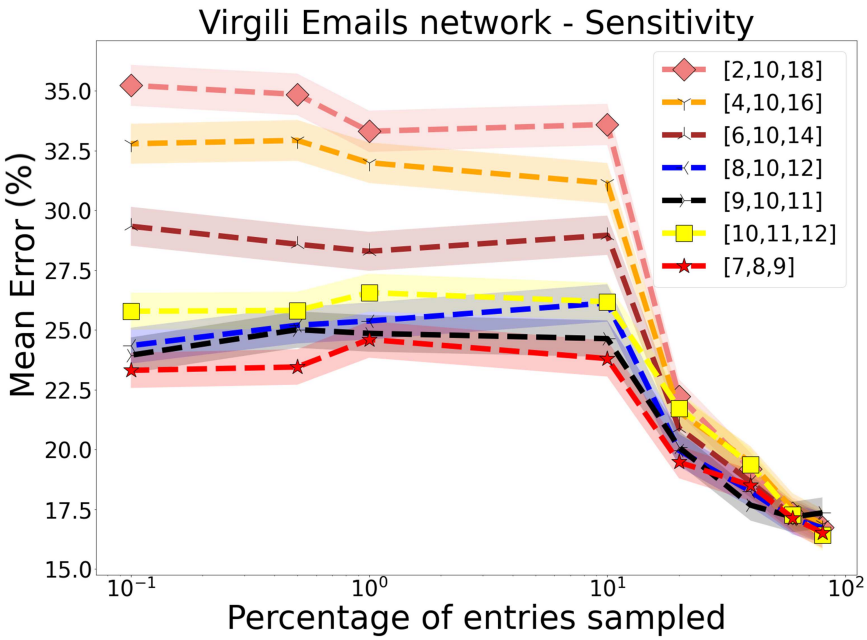
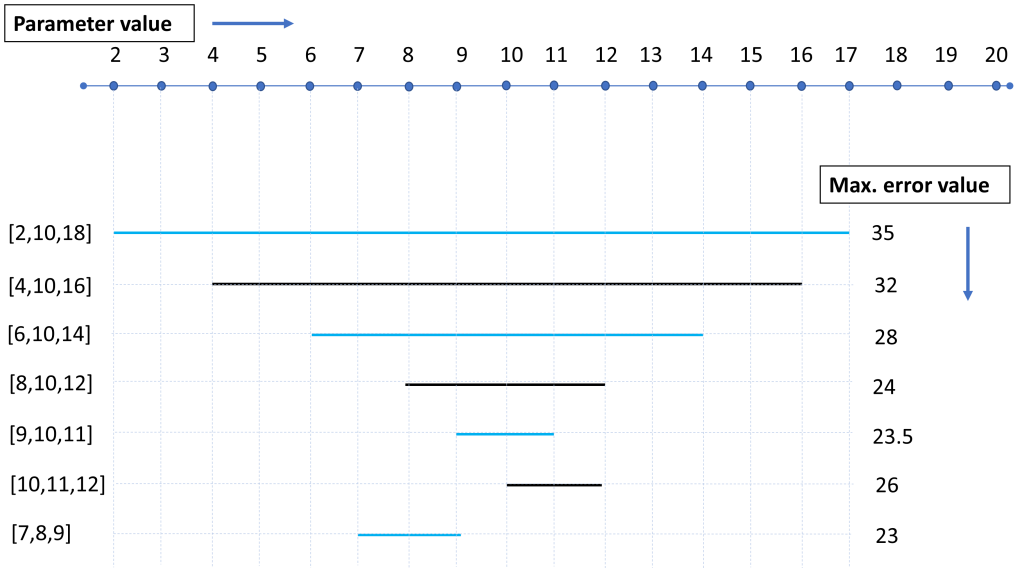


Figure 8.16: Bubble plot for Powerlaw 8 network: (a,b) comparison of Autoencoder trained on $m = [5,7,9]$ and (c,d) Matrix Completion for 40% and 99% missing entries respectively



(a)



(b)

Figure 8.17: Sensitivity analysis: The plots show that our model predicts more accurately when pre-trained on a more suitable and narrow range of pre-training parameters. More so, the accuracy is higher for parameters on the lower side of the average node degree as Power law networks have a right skewed node degree distribution with a high number of nodes with low node degree.

8.7 Discussion on optimum parameter selection for pre-training

This section furthers the intuition behind parameter selection for optimal pre-training when the average node degree of the test network is (tentatively) known.

We saw that the parameter m is analogous to node degree as it represents a number of edges attached to the new node while the artificial network is being constructed. We conduct a sensitivity analysis on the prediction performance throughout the parameter range. This helps us understand the model behavior and select the best training parameters for the given average node degree of the test network.

The mean error over the unobserved values of the Virgili Emails network, when autoencoder is pre-trained over various parameter windows, is shown in Fig. 8.17[a]. A more comparative visual display of the results can be seen in Fig. 8.17[b]. We can see that for an average node degree of 9.6, the autoencoder performs better when trained on narrower windows as compared to wider ones. Thus, the performance improves from [2,10,18] to [9,10,11]. We also observed that a narrow window placed on the lower side of the target node degree gives a slightly better performance than a window placed on the higher side of the spectrum, i.e., [7,8,9] performs better than [10,11,12], even if both the windows are equally narrow. This is because our target network is Power law in nature and a lot of nodes have degrees on the lower side of the average node degree.

8.8 Discussion on ensemble method

Deep learning allows us to learn complex, non-linear relationships in the data and offers increased flexibility. However, the model remains sensitive to training data and may find a different set of weights and hence different predictions every time they are trained. The stochastic nature of the learning algorithm means that the model might learn a slightly different version of the mapping function from input to output every time it is trained. This can be referred to as the variance in the model and a simple approach to reduce this variance is to train multiple models and combine the predictions from these models. This is called ensemble learning [226].

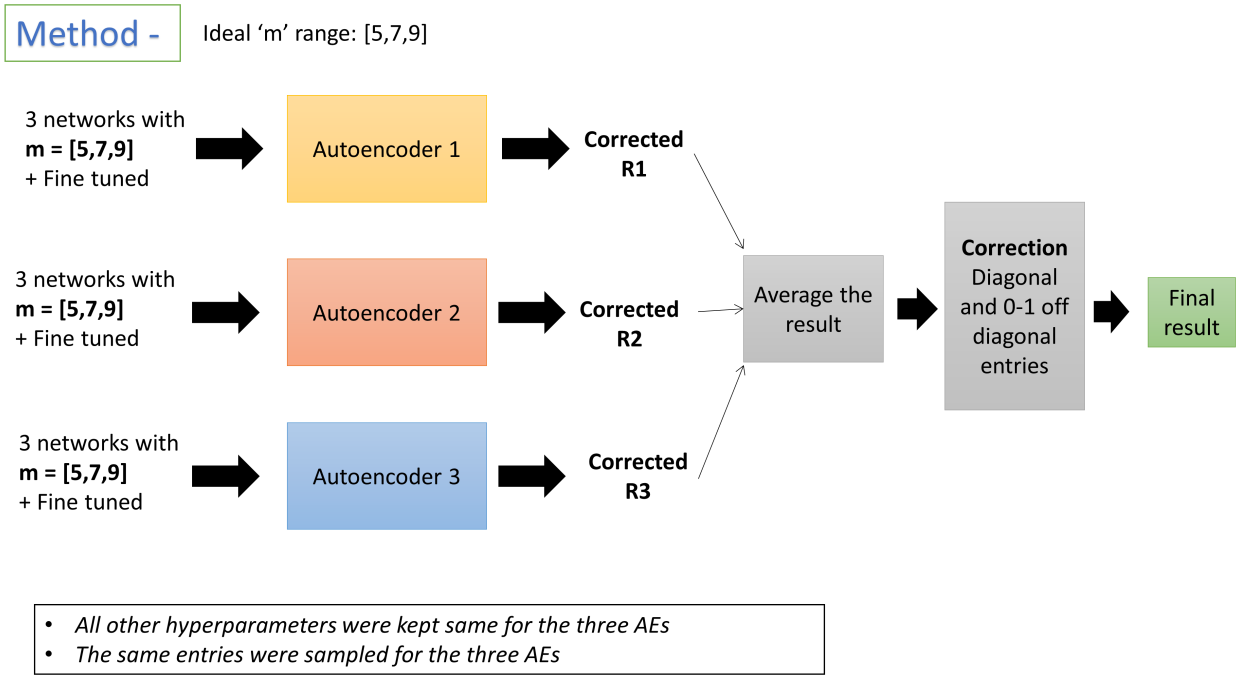


Figure 8.18: Model architecture used for ensemble approach. Three autoencoders are trained on the same parameters recommended by the Pre-training Oracle and the final result is averaged.

Ensembles were first introduced in [227–229]. This has proved helpful in improving prediction results for machine learning as well as deep learning models and since then there have been efforts to understand the optimal ensemble techniques and improve them [230–232]

As our experiments with ensemble learning at its initial stage, we started with using the simplest ensemble model architecture. As shown in Fig. 8.18, we train three learners on the parameter values (m) as recommended by our Pre-training Oracle, (OSP). All the hyperparameters for these learners are maintained the same, and thus the final result can be computed as the average of the result of each learner.

As we can see in Fig. 8.19, ensemble results (blue line with ☆ markers) show reduced errors over the fine tuned results (cyan line with ☆ markers) which are not ensembled. We did observe a slight improvement in the prediction performance of the pre-trained autoencoders when the results are averaged over three similar learners. These results can be further investigated to find the best number of learners, parameters they are trained on, and averaging method to achieve optimal performance for the given network. For instance, some of the possibilities can be:

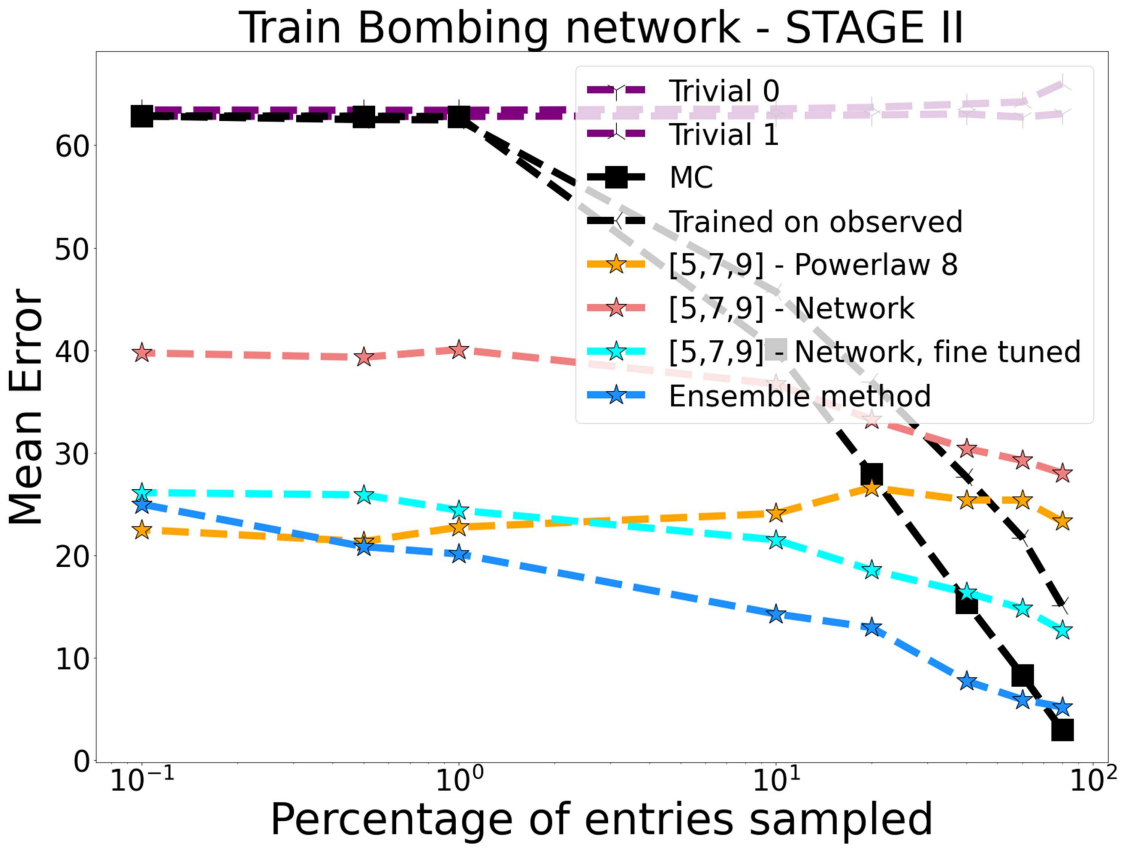


Figure 8.19: Prediction results for ensemble approach. The blue line shows that ensembling multiple autoencoders gives better prediction performance.

- different learners can be trained on different parameters values
- different number of synthetic networks, and
- the final result can be a weighted average where the weight can be determined by the performance of that learner

8.9 Conclusion

In this chapter, we have shown how to optimally pre-train a neural network when only sparse samples are available from the target network. We call this model "Oracle Search Pre-training" (OSP) as it helps us search the optimal pre-training parameter values specifically for the given social network. We use artificially generated data to compensate for the scarcity of real-world measurements to effectively train a neural network. The low-rankness of the distance matrices is leveraged.

Prediction performance is evaluated on three real-world networks, namely, Virgili Emails, Facebook, and Train Bombing network, under different variations of the model, and is also compared with a state-of-the-art Low-rank Matrix Completion approach. The model infers distances within one hop of the original value even when only 1% of measurements are sampled. We also study the sensitivity of the model towards the tuning parameter values and its relation with an average node degree of the social network. It is worth mentioning that though we illustrate results on social networks, the method can be generalized to other network domains by adapting the synthetic data.

In the future, the model can be tested on directed networks and networks from other domains such as criminal networks, transportation networks, etc. The effect of other network parameters, such as clustering coefficient, on reconstruction can also be investigated. Though we have used vanilla autoencoders in our work, the effect of using other autoencoders, such as Variational Autoencoders, would be interesting to study. Optimal ensemble architecture and subsequent improvement in the prediction performance is another branch of this experiment.

Chapter 9

Conclusion and Open Problems

A graph or a network is a collection of nodes and edges (links) connecting these nodes. Many real-world phenomenon correspond to graphs, and graphs are also indispensable in representing, modeling, and analyzing real-world systems that involve relationships or connections. Massive and complex graphs have become commonplace in the computing and communication infrastructure. Examples include social networks with users as nodes and their friendships as links, flight networks with airports as nodes and flight paths as edges, and the World Wide Web with webpages as nodes and hyperlinks as edges. With the ease of access to internet, increasing number of Wi-Fi-connected devices, increasing population, and content rich networks, we believe the networks we would be interested in analyzing in future would be even larger than what we have today. Cloud based services such as Azure, Google Cloud, and Amazon web services are adding to this infrastructure.

Analyzing graphs helps us to gain a deeper understanding of the structure, the relationships between nodes, and even the process of change in natural phenomenon such as human behavior, social interactions, and spread of an epidemic. Network science helped in answering critical questions and make impactful policies while tackling COVID-19 pandemic. Who is connecting to whom? Where are the connectors that are transmitting between communities? Where are the hub activities and how might the virus spread to other location? Which routes need to be closed to effectively control the spread between countries, states, and cities?

Many current network related algorithms and techniques that rely on the knowledge of the entire network or a major part thereof are becoming less and less applicable as the networks increase in size. Even techniques relying on sampling a significant fraction of a network may become expensive or impractical. Cost and complexity of capturing a network for analysis is a crucial factor when dealing with networks related to communication over a distributed structure, accessibility or inaccessibility of nodes, transfer of information to processing sites, and processing very large networks with computational constraints.

The focus of this research is on capturing and reconstructing the network topology from a very small set of distance measurements from the network. Additionally, the ability to estimate missing network measurements will aid in processing large networks that are only partially measured and in inferring characteristics of network which can help us make informed predictions.

The main contributions of this research focusing on inferring network topology from a very small set of distance measurements are as follows: (a) efficient representation of graphs for no loss of information, (b) predicting missing node-pair distances in large networks using Low-rank Matrix Completion, and (c) processing large graphs in the presence of partial measurements using neural networks.

(a) **Link Dimension:** Novel efficient graph modeling scheme using the concepts of "Link Dimension" and "Construction set" is proposed. Link dimension overcomes the limitations of metric dimension in reconstructing a graph from merely the distance vectors generated by resolution set by capturing node as well as edge information. This allows us to store a graph in a compact manner without losing any topology information. We derive link dimension for several graph families as well as relationship between various dimensions of a graph. An algorithm to find construction set from a given resolution set of a graph is also proposed.

(b) **Low-Rank Matrix Completion based Prediction:** We developed two schemes to estimate missing network distances from a small set of distance measurements. First scheme is based on deterministic technique of Low-Rank Matrix Completion (LMC). This is an optimization technique popular for completing partially observed low-rank matrices. LMC finds the best values of the missing entries under the constraint of maintaining a low-rank of the given matrix and given observed entries. We show that LMC accurately predicts missing distance entries in real-world undirected as well as directed networks (such as Facebook, Twitter, Email, and Blog networks) when we sample 40-60% of the network. The larger the network size, the lower sampling percentage can be handled by LMC. The computational complexity increases with increased missing entries. We also provide bounds for the missing distance entries to improve the accuracy of prediction further.

(c) **Neural Network based Prediction:** Our second prediction approach is based on neural networks. We were able to predict network distances even when the networks are ultra-sparsely sampled. This is achieved for the first time by training the autoencoder using similar but synthetically generated networks. We also propose a systematic method of selecting parameters of such synthetic training networks so as to keep the training data faithful to the real-world network that we need to predict. This is a novel approach of training neural networks for graph processing. The proposed model is able to train and predict network distances when 80% of the node-pair distances are known to even when only 1% of the node-pair distances are known while maintaining graceful degradation. This is very useful when the networks are large and the sampled measurements, even though large, constitute of only a very small part of the complete network.

With the concept of link dimension, many graphs can be represented in a lossless manner only by distances to a small set of nodes. While it is costly to find exact minimal construction set for large graphs, it may be possible to come up with a construction set or come close to one by sampling and verify how close it is. Being able to predict missing node-pair distances from partially observed distances enables us to sample and store smaller parts of vast networks and even make informed estimates about immeasurable distances such as probable distance between two terrorists.

The future of social and spatial networks is considered vast and distributed. We imagine an easy transfer, storage, convenient backups, and processing of such large networks with ease and speed at any point in the Internet of Things. Link dimension, predicting missing network distances, and being able to train a neural network to process real-world graphs even when complete networks are not sampled can help us achieve that. The concept of link dimension can facilitate data transfer in distributed systems with required low latency. Several existing analytics techniques rely on knowing all the entries. This either keeps us from analyzing networks that are only partially known or ignoring features that are not completely observed. Our techniques solve this problem by proposing a way to estimate missing entries that are in line with the network's originally assumed characteristics.

We would like to conclude this thesis with several open problems.

(a) **Link Dimension:** Modeling link dimension for other graph families would be an add-on to the exiting literature. Computing the construction set for large real-world networks using the proposed algorithm and analyzing its effect on existing storage techniques. We believe that the effectiveness of link dimension will be the largest for dynamic networks as the space saved for network storage will be multiplied by the number of frames required to store the evolution of the dynamic network. It would be interesting to see how network algorithms such as routing can be benefitted from such compact yet lossless network representation of spatial IoT networks.

(b) **Low-rank Matrix Completion based Prediction:** One can think of how can distance prediction be applied to dynamic networks. While we rely on the sparse decomposition of a given matrix H as L and S where L is the Low-rank component and S is the error, one can think of identifying anomalies from S and incorporating them to improve prediction as our given matrix H might not be exactly low in dimension. Extending the model for weighted graphs is also an interesting problem.

(c) **Neural Network-based Prediction:** Extending the proposed neural network model to directed networks would be the closest extension of the proposed scheme. An important improvement over the existing autoencoder based prediction model would be to remove the dependency of network architecture on the size of the network. We want to be able to fix the number of neurons in the input layer while being able to feed data from networks with different values of N, i.e., number of nodes on the networks. This can be achieved by: (a) feeding data in the format of $N \times f$ where f Is the number of features of the given node or (b) feeding data in the form of smaller square matrices of size $n \times n$ where $n << N$. We also imagined making a neural network black box that is trained for a certain family of graphs and can be used to predict missing network measurements for any graph that falls in that graph family. The black box could be available online for users to download and predict missing network measurements for the network of their choice without having the code or know the complexities of selecting the training parameters. This would provide a well-trained architecture for the given graph family and save the training process and time for

users. Furthermore, the scheme of pre-training neural networks to predict missing distances in networks can be investigated for its applicability to other social networks, and even other classes of networks that do not classify as scale-free.

Bibliography

[1] "WeRSM." https://wersm.com/how-much-data-is-generated-every-minute-on-social-media/, 2015.

[2] E. M. Daly and M. Haahr, "Social network analysis for information flow in disconnected delay-tolerant manets," *IEEE Transactions on Mobile Computing*, vol. 8, no. 5, pp. 606–621, 2008.

[3] G. Hua, Y. Sun, and D. Haughton, "Network analysis of us air transportation network," in *Data Mining for Social Network Data*, pp. 75–89, Springer, 2010.

[4] S. Wasserman and J. Galaskiewicz, *Advances in social network analysis: Research in the social and behavioral sciences*. Sage, 1994.

[5] S. Wasserman, K. Faust, *et al.*, *Social network analysis: Methods and applications*. Cambridge university press, 1994.

[6] S. P. Borgatti, A. Mehra, D. J. Brass, and G. Labianca, "Network analysis in the social sciences," *science*, vol. 323, no. 5916, pp. 892–895, 2009.

[7] A. Perliger and A. Pedahzur, "Social network analysis in the study of terrorism and political violence," *PS: Political Science and Politics*, vol. 44, no. 1, pp. 45–50, 2011.

[8] P. Domingos, "Mining social networks for viral marketing," *IEEE Intelligent Systems*, vol. 20, no. 1, pp. 80–82, 2005.

[9] N. K. Ahmed, J. Neville, and R. Kompella, "Network sampling: From static to streaming graphs," *ACM Trans. Knowl. Discov. Data*, vol. 8, pp. 7:1–7:56, June 2013.

[10] M. A. Cantrell and P. Lupinacci, "Methodological issues in online data collection," *Journal of advanced nursing*, vol. 60, no. 5, pp. 544–549, 2007.

[11] C. Hsu and B. A. Sandford, "The delphi technique: Making sense of consensus," *Practical Assessment Research and Evaluation*, vol. 12, no. 4, pp. 148–170, 2007.

[12] V. Cothey, "Web-crawling reliability," *Journal of the American Society for Information Science and Technology*, vol. 55, no. 14, pp. 1228–1238, 2004.

[13] P. Biernacki and D. Waldorf, "Snowball sampling: Problems and techniques of chain referral sampling," *Sociological Methods & Research*, vol. 10, no. 2, pp. 141–163, 1981.

[14] M. Sandelowski, "Combining qualitative and quantitative sampling, data collection, and analysis techniques in mixed-method studies," *Research in Nursing & Health*, vol. 23, no. 3, pp. 246–255, 2000.

[15] S. Edunov, C. G. Diuk, I. O. Filiz, S. Bhagat, and M. Burke, "Facebook structure." https://research.fb.com/three-and-a-half-degrees-of-separation/, 2016.

[16] "Ego Networks." http://analytictech.com/networks/egonet.htm.

[17] L. A. Goodman, "Snowball sampling," *The Annals of Mathematical Statistics*, vol. 32, no. 1, pp. 148–170, 1961.

[18] Wikipedia contributors, "Snowball sampling — Wikipedia, the free encyclopedia," 2018.

[19] P. A. Grabowicz, J. J. Ramasco, E. Moro, J. M. Pujol, and V. M. Eguiluz, "Social features of online networks: The strength of intermediary ties in online social media," *PLOS ONE*, vol. 7, pp. 1–9, 01 2012.

[20] G. Kossinets, "Effects of missing data in social networks," *Social Networks*, vol. 28, no. 3, pp. 247–268, 2006.

[21] R. Sharma, M. Magnani, and D. Montesi, "Investigating the types and effects of missing data in multilayer networks," in *IEEE/ACM Int. Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pp. 392–399, Aug 2015.

[22] S. Sahu, A. Mhedhbi, S. Salihoglu, J. Lin, and M. T. Özsu, "The ubiquity of large graphs and surprising challenges of graph processing: extended survey," *The VLDB Journal*, vol. 29, no. 2, pp. 595–618, 2020.

[23] L. B. Holder, R. Caceres, D. F. Gleich, J. Riedy, M. Khan, N. V. Chawla, R. Kumar, Y. Wu, C. Klymko, T. Eliassi-Rad, *et al.*, "Current and future challenges in mining large networks: Report on the second sdm workshop on mining networks and graphs," *SIGKDD Explor. Newsl.*, vol. 18, pp. 39–45, Aug. 2016.

[24] J. Fan, F. Han, and H. Liu, "Challenges of big data analysis," *National Science Review*, vol. 1, no. 2, pp. 293–314, 2014.

[25] A. Dasgupta, R. Kumar, and T. Sarlos, "On estimating the average degree," in *Proceedings of the 23rd International Conference on World Wide Web*, WWW '14, (New York, NY, USA), pp. 795–806, ACM, 2014.

[26] S. Hill, D. K. Agarwal, R. Bell, and C. Volinsky, "Building an effective representation for dynamic networks," *Journal of Computational and Graphical Statistics*, vol. 15, no. 3, pp. 584–608, 2006.

[27] "Scaling apache giraph to a trillion edges." https://www.facebook.com/notes/facebookengineering/scaling-apache-giraph-to-a-trillionedges/10151617006153920, 2013.

[28] P. Burkhardt and C. Waring, "An NSA Big Graph experiment," *Report NSA-RD-2013-056002v1*, vol. 29, no. 2, pp. 383–393, 2013.

[29] L. Dhulipala, G. E. Blelloch, and J. Shun, "Theoretically efficient parallel graph algorithms can be fast and scalable," *ACM Transactions on Parallel Computing (TOPC)*, vol. 8, no. 1, pp. 1–70, 2021.

[30] "Web data commons - hyperlink graphs." http://webdatacommons.org/hyperlinkgraph/.

[31] J. Tremblay, A. Prakash, D. Acuna, M. Brophy, V. Jampani, C. Anil, T. To, E. Cameracci, S. Boochoon, and S. Birchfield, "Training deep networks with synthetic data: Bridging the reality gap by domain randomization," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pp. 969–977, 2018.

[32] V. Allken, N. O. Handegard, S. Rosen, T. Schreyeck, T. Mahiout, and K. Malde, "Fish species identification using a convolutional neural network trained on synthetic data," *ICES Journal of Marine Science*, vol. 76, no. 1, pp. 342–349, 2019.

[33] M. Granovetter, "Network sampling: Some first steps," *American Journal of Sociology*, vol. 81, no. 6, pp. 1287–1303, 1976.

[34] W. contributors, "Sampling (statistics)." https://en.wikipedia.org/w/index.php?title=Sampling_(statistics)&oldid=855176039, 2018.

[35] A. Mislove, M. Marcon, K. P. Gummadi, P. Druschel, and B. Bhattacharjee, "Measurement and analysis of online social networks," in *Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement*, IMC '07, (New York, NY, USA), pp. 29–42, ACM, 2007.

[36] S. Ye, J. Lang, and F. Wu, "Crawling online social graphs," in *2010 12th International Asia-Pacific Web Conference*, pp. 236–242, April 2010.

[37] P. Biernacki and D. Waldorf, "Snowball sampling: using social networks to research non-heterosexual women," *International Journal of Social Research Methodology*, vol. 8, no. 1, pp. 47–60, 2005.

[38] J. Leskovec and J. J. Mcauley, "Learning to discover social circles in ego networks," in *Advances in Neural Information Processing Systems 25* (F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, eds.), pp. 539–547, Curran Associates, Inc., 2012.

[39] M. Gjoka, M. Kurant, C. T. Butts, and A. Markopoulou, "Walking in facebook: A case study of unbiased sampling of osns," in *2010 Proceedings IEEE Infocom*, pp. 1–9, Ieee, 2010.

[40] G. Cai, G. Lu, J. Guo, C. Ling, and R. Li, "Fast representative sampling in large-scale online social networks," *IEEE Access*, vol. 8, pp. 77106–77119, 2020.

[41] "Statistical Sampling." http://www.analysisgroup.com/practices/statistics-sampling/statistical-sampling/, 2018.

[42] J. Leskovec, J. Kleinberg, and C. Faloutsos, "Graph evolution: Densification and shrinking diameters," *ACM Trans. Knowl. Discov. Data*, vol. 1, Mar. 2007.

[43] F. Harary and R. A. Melter, "On the metric dimension of a graph," *Ars Combinatoria*, vol. 2, pp. 191–195, 1976.

[44] Q. C. Z. Kang, C. Peng, "Top-N recommender system via matrix completion," *Proc. 13th AAAI Conference on Artificial Intelligence*, 2016.

[45] G. Chartrand, L. Eroh, M. A. Johnson, and O. R. Oellermann, "Resolvability in graphs and the metric dimension of a graph," *Discrete Applied Mathematics*, vol. 105, no. 1, pp. 99 – 113, 2000.

[46] C. D. Dulanjalie and P. J. Anura, "Topology Preserving Maps From Virtual Coordinates for Wireless Sensor Networks," in *Proc. IEEE Local Computer Networks (LCN)*, (Oct), pp. 136–143, IEEE, oct 2010.

[47] Wikipedia, "Matrix completion." https://en.wikipedia.org/w/index.php?title=Matrix_completion&oldid=826222282, 2018.

[48] F. Masrour, I. Barjesteh, R. Forsati, A. Esfahanian, and H. Radha, "Network completion with node similarity: A matrix completion approach with provable guarantees," in *Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2015*, ASONAM '15, (New York, NY, USA), pp. 302–307, ACM, 2015.

[49] M. Kim and J. Leskovec, "The network completion problem: Inferring missing nodes and edges in networks," in *Proceedings of the 2011 SIAM International Conference on Data Mining*, pp. 47–58, SIAM, 2011.

[50] B. Recht, "A simpler approach to matrix completion.," *Journal of Machine Learning Research*, vol. 12, no. 12, 2011.

[51] E. J. Candès and Y. Plan, "Matrix completion with noise," *Proc. IEEE*, vol. 98, no. 6, p. 11, 2009.

[52] E. J. Candes and T. Tao, "The power of convex relaxation: Near-optimal matrix completion," *IEEE Transactions on Information Theory*, vol. 56, pp. 2053–2080, May 2010.

[53] R. H. Keshavan, A. Montanari, and S. Oh, "Matrix completion from a few entries," *IEEE Transactions on Information Theory*, vol. 56, pp. 2980–2998, June 2010.

[54] A. Annibale and A. C. C. Coolen, "What you see is not what you get: how sampling affects macroscopic features of biological networks," *Interface Focus*, vol. 1, no. 6, pp. 836–856, 2011.

[55] M. Papagelis, G. Das, and N. Koudas, "Sampling online social networks," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, pp. 662–676, March 2013.

[56] K. Anand, I. van Lelyveld, Á. Banai, S. Friedrich, R. Garratt, G. Hałaj, J. Fique, I. Hansen, S. M. Jaramillo, H. Lee, *et al.*, "The missing links: A global study on uncovering financial network structures from partial data," *Journal of Financial Stability*, vol. 35, pp. 107 – 119, 2018.

[57] A. De La Fuente, N. Bing, I. Hoeschele, and P. Mendes, "Discovery of meaningful associations in genomic data using partial correlation coefficients," *Bioinformatics*, vol. 20, no. 18, pp. 3565–3574, 2004.

[58] Y. R. Wang and H. Huang, "Review on statistical methods for gene network reconstruction using expression data," *Journal of Theoretical Biology*, vol. 362, pp. 53–61, 2014. Network-based biomarkers for complex diseases.

[59] M. A. Hasan and M. J. Zaki, *A Survey of Link Prediction in Social Networks*, pp. 243–275. Boston, MA: Springer US, 2011.

[60] D. Liben-Nowell and J. Kleinberg, "The link-prediction problem for social networks," *Journal of the American society for information science and technology*, vol. 58, no. 7, pp. 1019–1031, 2007.

[61] N. Benchettara, R. Kanawati, and C. Rouveirol, "Supervised machine learning applied to link prediction in bipartite social networks," in *2010 International Conference on Advances in Social Networks Analysis and Mining*, pp. 326–330, Aug 2010.

[62] H. H. Song, T. W. Cho, V. Dave, Y. Zhang, and L. Qiu, "Scalable proximity estimation and link prediction in online social networks," in *Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement*, IMC '09, pp. 322–335, ACM, 2009.

[63] J. Lee, H. Kim, J. Lee, and S. Yoon, "Transfer learning for deep learning on graph-structured data," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31, 2017.

[64] D. Duvenaud, D. Maclaurin, J. Aguilera-Iparraguirre, R. Gómez-Bombarelli, T. Hirzel, A. Aspuru-Guzik, and R. P. Adams, "Convolutional networks on graphs for learning molecular fingerprints," *arXiv preprint arXiv:1509.09292*, 2015.

[65] D. Boscaini, J. Masci, S. Melzi, M. M. Bronstein, U. Castellani, and P. Vandergheynst, "Learning class-specific descriptors for deformable shapes using localized spectral convolutional networks," in *Computer Graphics Forum*, vol. 34, pp. 13–23, Wiley Online Library, 2015.

[66] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," *arXiv preprint arXiv:1312.6203*, 2013.

[67] M. Henaff, J. Bruna, and Y. LeCun, "Deep convolutional networks on graph-structured data," *arXiv preprint arXiv:1506.05163*, 2015.

[68] L. Mou, G. Li, L. Zhang, T. Wang, and Z. Jin, "Convolutional neural networks over tree structures for programming language processing," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 30, 2016.

[69] G. Hinton, Y. LeCun, and Y. Bengio, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[70] M. Niepert, M. Ahmed, and K. Kutzkov, "Learning convolutional neural networks for graphs," in *International conference on machine learning*, pp. 2014–2023, PMLR, 2016.

[71] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[72] R. van den Berg, T. N. Kipf, and M. Welling, "Graph convolutional matrix completion," 2017.

[73] M. Mohammadrezaei, M. E. Shiri, and A. M. Rahmani, "Identifying fake accounts on social networks based on graph analysis and classification algorithms," *Security and Communication Networks*, vol. 2018, 2018.

[74] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training," 2018.

[75] D. Erhan, A. Courville, Y. Bengio, and P. Vincent, "Why does unsupervised pre-training help deep learning?," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 201–208, JMLR Workshop and Conference Proceedings, 2010.

[76] Y. Shang, W. Rumi, Y. Zhang, and M. Fromherz, "Localization from connectivity in sensor networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 15, pp. 961–974, Nov 2004.

[77] L. Getoor and C. P. Diehl, "Link mining: A survey," *SIGKDD Explor. Newsl.*, vol. 7, pp. 3–12, Dec. 2005.

[78] D. C. Dhanapala and A. P. Jayasumana, "Clueless nodes to network-cognizant smart nodes: Achieving network awareness in wireless sensor networks," in *Proc. IEEE Consumer Communications and Networking Conference (CCNC)*, pp. 174–179, Jan 2012.

[79] Z. Beerliova, F. Eberhard, T. Erlebach, A. Hall, M. Hoffmann, M. Mihal'ák, and L. S. Ram, "Network discovery and verification," *IEEE Journal on Selected Areas in Communications*, vol. 24, pp. 2168–2181, 2006.

[80] S. Khuller, B. Raghavachari, and A. Rosenfeld, "Localization in graphs," tech. rep., 1998.

[81] R. Manjusha and A. S. Kuriakose, "Metric dimension and uncertainty of traversing robots in anetwork," *International Journal on Applications of Graph Theory in Wireless Ad Hoc Networks and Sensor Networks (GRAPH-HOC)*, vol. 7, pp. 2–3, 2015.

[82] D. C. Dhanapala and A. P. Jayasumana, "Topology preserving maps—extracting layout maps of wireless sensor networks from virtual coordinates," *IEEE/ACM Transactions on Networking*, vol. 22, no. 3, pp. 784–797, 2013.

[83] G. A. Pendharkar and A. P. Jayasumana, *Virtual coordinate systems and coordinate-based operations for IoT*, ch. Performability in Internet of Things, Springer International (2019) 159–207., pp. 159–207. 2019.

[84] A. Caruso, S. Chessa, S. De, and A. Urpi, "Gps free coordinate assignment and routing in wireless sensor networks," in *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies.*, vol. 1, pp. 150–160, 2005.

[85] A. P. Jayasumana, R. Paffenroth, and S. Ramasamy, "Topology maps and distance-free localization from partial virtual coordinates for iot networks," in *Proc. IEEE International Conference on Communications (ICC)*, pp. 1–6, May (2016).

[86] D. C. Dhanapala and A. P. Jayasumana, "Anchor selection and topology preserving maps in WSNs - A Directional virtual coordinate based approach," in *Proceedings - Conference on Local Computer Networks, LCN*, pp. 571–579, 2011.

[87] A. P. Jayasumana, R. Paffenroth, G. Mahindre, S. Ramasamy, and K. Gajamannage, "Network topology mapping from partial virtual coordinates and graph geodesics," *IEEE/ACM Transactions on Networking*, vol. 27, no. 6, pp. 2405–2417, 2019.

[88] T. Bouchoucha, C. Chuah, and Z. Ding, "Finding link topology of large scale networks from anchored hop count reports," in *Proc. IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, Dec 2017.

[89] P. Slater, "Leaves of trees," *Congressus Numerantium*, vol. 14, pp. 549–559, 1975.

[90] J. Cáceres, C. Hernando, M. Mora, I. M. Pelayo, M. L. Puertas, C. Seara, and D. R. Wood, "On the metric dimension of some families of graphs," *Electronic Notes in Discrete Mathematics*, vol. 22, pp. 129–133, 2005. 7th International Colloquium on Graph Theory.

[91] D. C. Dhanapala and A. P. Jayasumana, "Directional virtual coordinate systems for wireless sensor networks," in *2011 IEEE International Conference on Communications (ICC)*, pp. 1–6, IEEE, 2011.

[92] A. Sebő and E. Tannier, "On metric generators of graphs," *Mathematics of Operations Research*, vol. 29, no. 2, pp. 383–393, 2004.

[93] R. C. Brigham, G. Chartrand, R. D. Dutton, and P. Zhang, "Resolving domination in graphs," *Mathematica Bohemica*, vol. 128, no. 1, pp. 25–36, 2003.

[94] M. A. Henning and O. R. Oellermann, "Metric-locating-dominating sets in graphs," *Ars Combinatoria*, vol. 73, no. 129-141, p. 94, 2004.

[95] G. Chartrand, V. Saenpholphat, and P. Zhang, "The independent resolving number of a graph," *Mathematica Bohemica*, vol. 128, no. 4, pp. 379–393, 2003.

[96] F. Okamoto, B. Phinezy, and P. Zhang, "The local metric dimension of a graph," *Mathematica Bohemica*, vol. 135, no. 3, pp. 239–255, 2010.

[97] I. G. Yero, A. Estrada-Moreno, and J. A. Rodrıguez-Velázquez, "The k-metric dimension of a graph: Complexity and algorithms," *arXiv preprint arXiv:1401.0342*, 2014.

[98] I. G. Yero, M. Jakovac, D. Kuziak, and A. Taranenko, "The partition dimension of strong product graphs and cartesian product graphs," *Discrete Mathematics*, vol. 331, pp. 43–52, 2014.

[99] A. Kelenc, N. Tratnik, and I. G. Yero, "Uniquely identifying the edges of a graph: the edge metric dimension," *Discrete Applied Mathematics*, vol. 251, pp. 204–220, 2018.

[100] J. Kratica, "Strong metric dimension: A survey," *Yugoslav Journal of Operations Research*, vol. 24, pp. 187–198, 2014.

[101] J. Cáceres, C. Hernando, M. Mora, I. M. Pelayo, M. L. Puertas, C. Seara, and D. R. Wood, "On the metric dimension of cartesian products of graphs," *SIAM Journal on Discrete Mathematics*, vol. 21, no. 2, pp. 423–441, 2007.

[102] E. W. Weisstein, ""Grid Graph" From MathWorld." Accessed: 2021-07-31.

[103] C. X. Kang and E. Yi, "The fractional strong metric dimension of graphs," in *International Conference on Combinatorial Optimization and Applications*, pp. 84–95, Springer, 2013.

[104] O. R. Oellermann and J. Peters-Fransen, "The strong metric dimension of graphs and digraphs," *Discrete Applied Mathematics*, vol. 155, no. 3, pp. 356 – 364, 2007.

[105] O. R. Oellermann and J. Peters-Fransen, "The strong metric dimension of graphs and digraphs," *Discrete Applied Mathematics*, vol. 155, no. 3, pp. 356 – 364, 2007.

[106] D. Pedoe, *Geometry: A comprehensive course*. Courier Corporation, 2013.

[107] D. M. Cvetkovic and I. Gutman, *Applications of graph spectra*. Matematički institut SANU, 2009.

[108] D. Cvetković and M. Petrić, "A table of connected graphs on six vertices," *Discrete Mathematics*, vol. 50, pp. 37–49, 1984.

[109] R. Balakrishnan, "The energy of a graph," *Linear Algebra and its Applications*, vol. 387, pp. 287 – 295, 2004.

[110] M. Morzy and T. Kajdanowicz, "Graph energies of egocentric networks and their correlation with vertex centrality measures," *Entropy*, vol. 20, no. 12, p. 916, 2018.

[111] J. Zhang and I. C. Paschalidis, "Statistical anomaly detection via composite hypothesis testing for Markov models," *IEEE Transactions on Signal Processing*, vol. 66, pages 589–602, Feb 2018.

[112] J. Zhang and I. C. Paschalidis, "An improved composite hypothesis test for Markov models with applications in network anomaly detection," in *Proc. 54th IEEE Conference on Decision and Control (CDC)*, pp. 3810–3815, Dec 2015.

[113] "Network Science (NRC)," *Report by Committee on Network Science for Future Army Applications*, 2005.

[114] T. Le and N. Ono, "Closed-form and near closed-form solutions for TDOA-based joint source and sensor localization," *IEEE Transactions on Signal Processing*, vol. 65, pages 1207–1221, Mar 2017.

[115] H. A. Oliveira, E. F. Nakamura, A. A. Loureiro, and A. Boukerche, "Error analysis of localization systems for sensor networks," in *Proc. 13th annual ACM International Workshop on Geographic Information Systems*, pp. 71–78, ACM, 2005.

[116] Q. Cao and T. Abdelzaher, "Scalable logical coordinates framework for routing in wireless sensor networks," in *ACM Transactions on Sensor Networks,Vol. 2,pp. 557-593, Nov 2006.*, 2006.

[117] A. Caruso, S. Chessa, S. De, and A. Urpi, "GPS free coordinate assignment and routing in wireless sensor networks," in *Proc. INFOCOM*, 2005.

[118] K. Liu and N. Abu-Ghazaleh, "Aligned virtual coordinates for greedy routing in WSNs," in *Proc. IEEE Int. Conf. on Mobile Ad Hoc and Sensor Systems (MASS)*, pp. 377–386, 2007.

[119] A. Rao, S. Ratnasamy, C. Papadimitriou, S. Shenker, and I. Stoica, "Geographic routing without location information," in *Proceedings of the 9th annual international conference on Mobile computing and networking*, pp. 96–108, ACM, 2003.

[120] D. Dhanapala and A. Jayasumana, "CSR: Convex subspace routing protocol for wsns," in *Proc. IEEE Conference on Local Computer Networks (LCN)*, 2009.

[121] Z. Lin, M. Chen, and Y. Ma, "The Augmented Lagrange Multiplier Method for Exact Recovery of Corrupted Low-Rank Matrices," *arXiv:1009.5055*, p. 23, 2013.

[122] E. J. Candès and B. Recht, "Exact matrix completion via convex optimization," *Foundations of Computational mathematics*, vol. 9, no. 6, p. 717, 2009.

[123] R. Paffenroth, P. du Toit, R. Nong, L. Scharf, A. P. Jayasumana, and V. Bandara, "Space-time signal processing for distributed pattern detection in sensor networks," *IEEE Journal of Selected Topics in Signal Processing*, vol. 7, no. 1, pp. 38–49, 2013.

[124] D. C. Dhanapala and A. P. Jayasumana, "Geo-logical routing in wireless sensor networks," in *2011 8th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks*, pp. 305–313, IEEE, jun 2011.

[125] D. C. Dhanapala, a. P. Jayasumana, and S. Mehta, "On Boundary Detection of 2-D and 3-D Wireless Sensor Networks," *2011 IEEE Global Telecommunications Conference - GLOBECOM 2011*, pp. 1–5, 2011.

[126] S. Chatterjee, "Matrix estimation by universal singular value thresholding," *Ann. Statist.*, vol. 43, pages 177–214, 2015.

[127] J. B. Tenenbaum, V. De Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, no. 5500, pp. 2319–2323, 2000.

[128] J. A. Lee and M. Verleysen, *Nonlinear Dimensionality Reduction*. Springer, 2007.

[129] R. C. Paffenroth, R. Nong, and P. C. D. Toit, "On covariance structure in noisy, big data," *Proc. SPIE*, vol. 8857, 2013.

[130] J. B. Kruskal, "Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis," *Psychometrika*, vol. 29, pages 1–27, 1964.

[131] I. Borg and P. Groenen, *Modern Multidimensional Scaling - Theory and Applications*. Springer New York, 2005.

[132] B. Eriksson, P. Barford, and R. Nowak, "Estimating hop distance between arbitrary host pairs," in *IEEE INFOCOM 2009*, pp. 801–809, IEEE, 2009.

[133] K. Chou, "Applications of graph theory to enzyme kinetics and protein folding kinetics: Steady and non-steady-state systems," *Biophysical Chemistry*, vol. 35, pages 1–24, 1990.

[134] J. T. L. Wang, M. J. Zaki, H. T. T. Toivonen, and D. Shasha, *Introduction to Data Mining in Bioinformatics*, pp. 3–8. Springer London, 2005.

[135] R. Pech, L. P. D. Hao, and T. Z. H. Cheng, "Link prediction via matrix completion," *EPL (Europhysics Letters)*, vol. 117, no. 2, 2017.

[136] S. Al-Homidan, H. Wolkowicz, S. Al-Homidan, and H. Wolkowicz, "Approximate and exact completion problems for Euclidean distance matrices using semidefinite programming," *Linear Algebra and its Applications*, vol. 406, pp. 109–141, sep 2005.

[137] B. Mishra, G. Meyer, and R. Sepulchre, "Low-rank optimization for distance matrix completion," *Proc. IEEE Conference on Decision and Control*, pp. 4455–4460, 2011.

[138] N. Krislock, H. Wolkowicz, N. Krislock, and H. Wolkowicz, *Euclidean distance matrices and applications*. Springer, 2012.

[139] A. Y. Alfakih, "Graph rigidity via Euclidean distance matrices," *Linear Algebra and Its Applications*, vol. 310, pp. 149–165, 2000.

[140] J. C. Gower, "Euclidean distance geometry," *Mathematical Scientist*, vol. 7, pages 1–14, 1982.

[141] Y. S. T. L. N. Nguyen, "Matrix completion optimization for localization in wireless sensor networks for intelligent IoT," *Sensors*, vol. 16, 2016.

[142] M. Abramowitz and I. A. Stegun, *Handbook of Mathematical Functions with Formulas, Grpahs and Mathematical Tables*, p. 300. 1964.

[143] R. Diestel, *Graph Theory*, vol. 173. Springer Verlag, 2005.

[144] K. Mehlhorn and P. Sanders, *Algorithms and Data Structures: The Basic Toolbox*. 2008.

[145] C. Eckart and G. Young, "The approximation of one matrix by another of lower rank," *Psychometrika*, vol. 1, pp. 211–218, 1936.

[146] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer New York, vol. 4, 2006.

[147] Q. Wu, C. L. Lu, and R. C. T. Lee, "The approximability of the weighted Hamiltonian path completion problem on a tree," *Theoretical Computer Science*, vol. 341, pages 385–397, 2005.

[148] D. Gamarnik and M. Sviridenko, "Hamiltonian completions of sparse random graphs," *Discrete Applied Mathematics*, vol. 152, pages 139–158, 2005.

[149] A. Parra and P. Scheffler, "Characterizations and algorithmic applications of chordal graph embeddings," *Discrete Applied Mathematics*, vol. 79, pages 171–188, 1997.

[150] I. Gutman and B. Borovicanin, "Nullity of graphs: an updated survey," *Selected topics on applications of graph spectra, Math. Inst., Belgrade*, pp. 137–154, 2011.

[151] F. R. K. Chung, *Spectral Graph Theory (CBMS Regional Conference Series in Mathematics)*. American Mathematical Society, 1996.

[152] M. Aouchiche and P. Hansen, "Distance spectra of graphs: A survey," vol. 458, pages 301–386, Oct 2014.

[153] P. Holme and B. J. Kim, "Growing scale-free networks with tunable clustering," *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, vol. 65, 2002.

[154] A. Barabasi and R. Albert, "Emergence of scaling in random networks," *Science (New York, N.Y.)*, vol. 286, pages 509–12, Oct 1999.

[155] J. Leskovec and A. Krevl, "SNAP Datasets: Stanford Large Network Dataset Collection," Jun 2014.

[156] A. F. Buoud and A. P. Jayasumana, "Topology preserving map to physical map - a thin-plate spline based transform," in *Proc. IEEE 41st Conference on Local Computer Networks (LCN)*, pp. 262–270, Nov 2016.

[157] S. Diamond, E. Chu, and S. Boyd, "CVXPY: A Python-embedded modeling language for convex optimization, version 0.2," May 2014.

[158] G. V. Rossum and F. L. Drake, "Python Tutorial," *History*, vol. 42, pp. 1–122, 2010.

[159] T. B. Hashimoto, Y. Sun, and T. S. Jaakkola, "Metric recovery from directed unweighted graphs,"

[160] R. Paffenroth, "Matrix Completion Algorithm," 2017. Personal communication.

[161] N. Al-nabhan, A. B. Othman, A. Jayasumana, and M. Gunjan, "Directional virtual coordinate approach for 3-d iot systems in smart cities," in *International Conference on Big Data and Security*, pp. 479–498, Springer, 2019.

[162] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: Bringing order to the web.," tech. rep., Stanford InfoLab, 1999.

[163] J. Ugander, B. Karrer, L. Backstrom, and C. Marlow, "The Anatomy of the Facebook Social Graph," *CoRR*, vol. abs/1111.4, 2011.

[164] M. Cha, A. Mislove, and K. P. Gummadi, "A Measurement-driven Analysis of Information Propagation in the Flickr Social Network," in *Proceedings of the 18th International Conference on World Wide Web*, WWW '09, (New York, NY, USA), pp. 721–730, ACM, 2009.

[165] S. A. Catanese, P. De Meo, E. Ferrara, G. Fiumara, and A. Provetti, "Crawling Facebook for social network analysis purposes," in *Proc. International Conference on Web Intelligence, Mining and Semantics*, WIMS '11, pp. 52:1–52:8, ACM, 2011.

[166] S. Chakrabarti, M. van den Berg, and B. Dom, "Focused crawling: a new approach to topic-specific web resource discovery," *Computer Networks*, vol. 31, pages 1623–1640, 1999.

[167] M. Bakonyi and C. R. Johnson, "The Euclidian Distance Matrix Completion Problem," *SIAM Journal on Matrix Analysis and Applications*, vol. 16, pp. 646–654, apr 1995.

[168] M. W. Trosset, "Distance Matrix Completion by Numerical Optimization," *Computational Optimization and Applications*, vol. 17, no. 1, pp. 11–22, 2000.

[169] D. C. Dhanapala and A. P. Jayasumana, "Network-Awareness via Self-Learning at Wireless Sensor Nodes," in *IEEE Transactions on Parallel and Distributed Systems*, 2011.

[170] D. A. Spielman, "Spectral Graph Theory," *in Combinatorial Scientific Computing*, p. Chapter 6, 2012.

[171] F. R. K. Chung, "Spectral Graph Theory," *ACM SIGACT News*, vol. 30, p. 14, 1999.

[172] R. Kannan and S. Vempala, "Spectra of Graphs," *Foundations and Trends in Theoretical Computer Science*, vol. 4, no. 3-4, pp. 132–288, 2008.

[173] K. Börner, S. Sanyal, and A. Vespignani, "Network science.," *Annual Review of Information Science & Technology*, vol. 41, pp. 537–607, 2007.

[174] L. Backstrom and J. Leskovec, "Supervised Random Walks: Predicting and Recommending Links in Social Networks," in *Proceedings of the Fourth ACM International Conference on Web Search and Data Mining*, WSDM '11, (New York, NY, USA), pp. 635–644, ACM, 2011.

[175] S. Catanese, P. D. Meo, E. Ferrara, G. Fiumara, and A. Provetti, "Crawling Facebook for Social Network Analysis Purposes," *CoRR*, vol. abs/1105.6, 2011.

[176] C. Wong, K. Wong, K. Ng, W. Fan, and K. Yeung, "Design of a Crawler for Online Social Networks Analysis," in *WSEAS Transactions on Communications*, 2014.

[177] J. Travers, S. Milgram, J. Travers, and S. Milgram, "An Experimental Study of the Small World Problem," *Sociometry*, vol. 32, pp. 425–443, 1969.

[178] R. Paffenroth, R. Nong, and P. D. Toit, "Fast Alternating Direction Method of Multipliers Solvers for Robust Principal Component Analysis and Big Data," *in preparation*, 2013.

[179] J. Leskovec and A. Krevl, "SNAP Datasets: Stanford large network dataset collection." http://snap.stanford.edu/data, June 2014.

[180] "Clustering coefficient - wikipedia." https://en.wikipedia.org/wiki/Clustering_coefficient.

[181] A. Hashmi, F. Zaidi, A. Sallaberry, and T. Mehmood, "Are all social networks structurally similar?," in *Proc. 2012 Int. Conference on Advances in Social Networks Analysis and Mining (ASONAM 2012)*, pp. 310–314, IEEE Computer Society, 2012.

[182] "Network science - Wikipedia." https://en.wikipedia.org/wiki/Network_science, April 2021.

[183] J. M. Joyce, *Kullback-Leibler Divergence*, pp. 720–722. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011.

[184] D. Duan, Y. Li, Y. Jin, and Z. Lu, "Community mining on dynamic weighted directed graphs," in *Proc. 1st ACM Int. Workshop on Complex Networks Meet Information& Knowledge Management*, CNIKM, pp. 11–18, 2009.

[185] P. Gai and S. Kapadia, "Contagion in financial networks," in *Proc. R. Soc. A*, March 2010.

[186] B. W. Hung, A. P. Jayasumana, and V. W. Bandara, "INSiGHT: A system to detect violent extremist radicalization trajectories in dynamic graphs," *Data & Knowledge Engineering*, vol. 118, pp. 52 – 70, 2018.

[187] L. A. Meyers, M. Newman, and B. Pourbohloul, "Predicting epidemics on directed contact networks," *Jour. of Theoretical Biology*, vol. 240, no. 3, pp. 400–418, 2006.

[188] J. M. Kleinberg, "Challenges in mining social network data: Processes, privacy, and paradoxes," in *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '07, (New York, NY, USA), pp. 4–5, ACM, 2007.

[189] S. Fortunato, "Community detection in graphs," *Physics Reports*, vol. 486, no. 3, pp. 75 – 174, 2010.

[190] J. Leskovec and C. Faloutsos, "Sampling from large graphs," in *Proc. 12th ACM SIGKDD Int. Conference on Knowledge Discovery and Data Mining*, KDD, pp. 631–636, ACM, 2006.

[191] J. H. Koskinen, G. L. Robins, P. Wang, and P. E. Pattison, "Bayesian analysis for partially observed network data, missing ties, attributes and actors," *Social Networks*, vol. 35, no. 4, pp. 514 – 527, 2013.

[192] J. Huang, F. Nie, H. Huang, Y. Lei, and C. H. Q. Ding, "Social trust prediction using rank-k matrix recovery," in *IJCAI*, 2013.

[193] S. Chen, A. Sandryhaila, J. M. F. Moura, and J. Kovačević, "Signal recovery on graphs: Variation minimization," *IEEE Transactions on Signal Processing*, vol. 63, pp. 4609–4624, Sep. 2015.

[194] M. Studený, "A recovery algorithm for chain graphs," *Int. Jour. of Approximate Reasoning*, vol. 17, no. 2, pp. 265 – 293, 1997.

[195] T. A. Snijders and K. Nowicki, "Estimation and prediction for stochastic blockmodels for graphs with latent block structure," *Jour. of Classification*, vol. 14, pp. 75–100, Jan 1997.

[196] V. J. Barranca, D. Zhou, and D. Cai, "Low-rank network decomposition reveals structural characteristics of small-world networks," *Phys. Rev. E*, vol. 92, p. 062822, Dec 2015.

[197] J. Leskovec and C. Faloutsos, "Sampling from large graphs," in *Proc. 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 631–636, ACM, 2006.

[198] A. Awan, R. A. Ferreira, S. Jagannathan, and A. Grama, "Distributed uniform sampling in unstructured peer-to-peer networks," in *Proceedings of the 39th Annual Hawaii International Conference on System Sciences (HICSS'06)*, vol. 9, pp. 223c–223c, IEEE, 2006.

[199] "KONECT Datasets." http://konect.uni-koblenz.de/, April 2017.

[200] M. He and S. Petoukhov, *Mathematics of bioinformatics: Theory, methods and applications*, vol. 19. John Wiley and Sons, 2011.

[201] D. A. Luke and J. K. Harris, "Network analysis in public health: history, methods, and applications," *Annu. Rev. Public Health*, vol. 28, pp. 69–93, 2007.

[202] W. Wei, J. Erenrich, and B. Selman, "Towards efficient sampling: Exploiting random walk strategies," in *AAAI*, vol. 4, pp. 670–676, 2004.

[203] R. DeJordy and D. Halgin, "Introduction to ego network analysis," *Boston MA: Boston College and the Winston Center for Leadership & Ethics*, 2008.

[204] J. A. Smith, J. Moody, and J. H. Morgan, "Network sampling coverage II: The effect of non-random missing data on network measurement," *Social networks*, vol. 48, pp. 78–99, 2017.

[205] G. Mahindre, A. P. Jayasumana, K. Gajamannage, and R. Paffenroth, "On sampling and recovery of topology of directed social networks – a low-rank matrix completion based approach," in *2019 IEEE 44th Conference on Local Computer Networks (LCN)*, pp. 324–331, 2019.

[206] B. Mehta, M. Diaz, F. Golemo, C. J. Pal, and L. Paull, "Active domain randomization," *arXiv preprint arXiv:1904.04762*, 2019.

[207] S. De Moor, C. Vandeviver, and T. Vander Beken, "Assessing the missing data problem in criminal network analysis using forensic dna data," *Social Networks*, vol. 61, pp. 99–106, 2020.

[208] G. Udny Yule, "A mathematical theory of evolution, based on the conclusions of Dr. JC Willis, FRS," *RSPTB*, vol. 213, pp. 21–87, 1925.

[209] A. Barabási *et al.*, *Network science*. Cambridge University Press, 2016.

[210] C. Zang, P. Cui, C. Faloutsos, and W. Zhu, "On power law growth of social networks," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, pp. 1727–1740, sep 2018.

[211] R. Albert and A. Barabási, "Statistical mechanics of complex networks," *Reviews of modern physics*, vol. 74, no. 1, p. 47, 2002.

[212] E. M. Jin, M. Girvan, and M. E. Newman, "Structure of growing social networks," *Physical review E*, vol. 64, no. 4, p. 046132, 2001.

[213] C. Zhou and R. C. Paffenroth, "Anomaly detection with robust deep autoencoders," in *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 665–674, 2017.

[214] A. A. Hagberg, D. A. Schult, and P. J. Swart, "Exploring network structure, dynamics, and function using networkx," in *Proceedings of the 7th Python in Science Conference* (G. Varoquaux, T. Vaught, and J. Millman, eds.), (Pasadena, CA USA), pp. 11 – 15, 2008.

[215] R. Karkare, G. Mahindre, C. Zhou, and R. Paffenroth, "Hadamard Autoencoders for Inference in Ultra-Sparse Data," July 2020.

[216] P. Sarkar and A. W. Moore, "Dynamic social network analysis using latent space models," *ACM SIGKDD Explorations Newsletter*, vol. 7, no. 2, pp. 31–40, 2005.

[217] M. Boguná, R. Pastor-Satorras, A. Díaz-Guilera, and A. Arenas, "Models of social networks based on social distance attachment," *Physical review E*, vol. 70, no. 5, p. 056122, 2004.

[218] G. Mahindre, A. P. Jayasumana, K. Gajamannage, and R. Paffenroth, "On sampling and recovery of topology of directed social networks–a low-rank matrix completion based approach," in *2019 IEEE 44th Conference on Local Computer Networks (LCN)*, pp. 324–331, IEEE, 2019.

[219] J. Qi, W. Wang, R. Zhang, and Z. Zhao, "A learning based approach to predict shortest-path distances," 2020.

[220] W. Liu, Y. Jia, G. Jiang, H. Jiang, F. Wu, and Z. Lv, "Wifi-sensing based person-to-person distance estimation using deep learning," in *2018 IEEE 24th International Conference on Parallel and Distributed Systems (ICPADS)*, pp. 236–243, IEEE, 2018.

[221] G. Mahindre, R. Karkare, R. Paffenroth, and A. Jayasumana, "Inference in social networks from ultra-sparse distance measurements via pretrained hadamard autoencoders," in *2020 IEEE 45th Conference on Local Computer Networks (LCN)*, pp. 256–266, IEEE, 2020.

[222] R. Karkare, R. Paffenroth, and G. Mahindre, "Blind image denoising and inpainting using robust hadamard autoencoders," 2021.

[223] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. icml*, vol. 30, p. 3, Citeseer, 2013.

[224] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Icml*, 2010.

[225] R. Hecht-Nielsen, "Theory of the backpropagation neural network," in *Neural networks for perception*, pp. 65–93, Elsevier, 1992.

[226] J. Brownlee, "Ensemble Learning Methods for Deep Learning Neural Networks," August 2019.

[227] L. Hansen and P. Salamon, "Neural network ensembles," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 10, pp. 993–1001, 1990.

[228] M. P. Perrone and L. N. Cooper, "When networks disagree: Ensemble methods for hybrid neural networks," tech. rep., BROWN UNIV PROVIDENCE RI INST FOR BRAIN AND NEURAL SYSTEMS, 1992.

[229] A. Krogh and J. Vedelsby, "Neural network ensembles, cross validation and active learning," in *Proceedings of the 7th International Conference on Neural Information Processing Systems*, NIPS'94, (Cambridge, MA, USA), p. 231–238, MIT Press, 1994.

[230] W. Li and R. Paffenroth, "Optimal ensembles for deep learning classification: Theory and practice," in *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*, pp. 1658–1665, IEEE, 2019.

[231] G. Huang, Y. Li, G. Pleiss, Z. Liu, J. E. Hopcroft, and K. Q. Weinberger, "Snapshot ensembles: Train 1, get m for free," *arXiv preprint arXiv:1704.00109*, 2017.

[232] J. Xie, B. Xu, and Z. Chuang, "Horizontal and vertical ensemble with deep representation for classification," *arXiv preprint arXiv:1306.2759*, 2013.

[233] S. Khuller, B. Raghavachari, and A. Rosenfeld, "Landmarks in graphs," *Discrete Applied Mathematics*, vol. 70, no. 3, pp. 217–229, 1996.

[234] A. K. Menon and C. Elkan, "Link prediction via matrix factorization," in *Joint european conference on machine learning and knowledge discovery in databases*, pp. 437–452, Springer, 2011.

[235] J. J. Wang, H. Bensmail, and X. Gao, "Multiple graph regularized nonnegative matrix factorization," *Pattern Recognition*, vol. 46, no. 10, pp. 2840–2847, 2013.

[236] J. Leskovec, D. Chakrabarti, J. Kleinberg, C. Faloutsos, and Z. Ghahramani, "Kronecker graphs: an approach to modeling networks.," *Journal of Machine Learning Research*, vol. 11, no. 2, 2010.

Appendix A

Tools and datasets

1. Extracting information from partial data

   Tools: MATLAB , Python

   Dataset: SNAP dataset

2. Link prediction using Virtual Coordinates

   Tools: MATLAB, Python

   Dataset: CNRL dataset, KONECT dataset, SNAP dataset

Appendix B

License

Colorado State University LaTeX Thesis Template

by Elliott Forney – 2017

This is free and unencumbered software released into the public domain.

Anyone is free to copy, modify, publish, use, compile, sell, or distribute this software, either in source code form or as a compiled binary, for any purpose, commercial or non-commercial, and by any means.

In jurisdictions that recognize copyright laws, the author or authors of this software dedicate any and all copyright interest in the software to the public domain. We make this dedication for the benefit of the public at large and to the detriment of our heirs and successors. We intend this dedication to be an overt act of relinquishment in perpetuity of all present and future rights to this software under copyright law.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.