Research in Industrial Projects for Students

Sponsor

RAND Corporation

Final Report

Deep Generative Approaches to Network Science for Social System Simulations

Student Members

Amiri Hayes (Project Manager), *New Jersey Institute of Technology*

`rips2024_rand@ipam.ucla.edu`

Mariia Sinkevich, *University of Oxford*

Miontranese Green, *California State University, Long Beach*

Siying Ding, *Barnard College*

Academic Mentor

Yuanzhou Adrian Chen, `adrianchen@cs.ucla.edu`

Sponsoring Mentors

Dr. Raffaele Vardavas, `rvardava@rand.org`
Dr. Ben Gibson, `bgibson@rand.org`
Dr. Prateek Puri, `ppuri@rand.org`

Date: August 21, 2024

This project was jointly supported by RAND Corporation and NSF Grant DMS 1925919.

Abstract

Microsimulation and agent-based models are increasingly being used to simulate complex social systems and to understand everyday interaction networks. However, accurate microsimulation models require comprehensive network data sets that combine large-scale network structures with detailed individual-level behavioral characteristics. Integrating diverse network data sources to produce accurate simulation models of complex social phenomena is a challenging task with limited research. Our project focuses on two main objectives:

1. Generate synthetic networks that are statistically and structurally equivalent to an existing sociocentric network dataset from Portland, Oregon. We used the Iterative Local Expansion method to achieve a scalable machine learning model for graph generation.

2. Perform network fusion by combining the synthetically generated dataset with egocentric network survey data to create a comprehensive social network. To achieve this goal, we developed a mathematical algorithm that accounts for egocentric node features and models the time-dependent behavior of the sociocentric network through node interactions.

The ultimate aim of this project is to further researcher's ability to generative complex and comprehensive graph network datasets for populations of interest for the purpose of improving social simulations like those in disease transmission.

Acknowledgments

We would like to express our gratitude to the Research In Industrial Projects (RIPS) program at the Institute for Pure and Applied Mathematics (IPAM) at UCLA for providing us with this valuable research opportunity. We thank our academic mentor, Adrian Chen, for his guidance in formulating our problems, advancing our research, and directing us towards more effective methodologies, which have improved the quality of our work. We are also grateful to our industry mentor, Dr. Raffaele Vardavas, for his extensive knowledge and provision of references, which have given us a better understanding of the context and origins of our project. Additionally, we appreciate the support, organization, and dedication of all the staff at IPAM in fostering a collaborative research environment. Without the combined efforts of the RIPS program, mentors, and staff, this research would not have been possible. Thank you for your support and encouragement.

Contents

List of Figures

Chapter 1

Introduction

The RAND Corporation has long been at the forefront of research in various fields, including health, defense, and technology. Founded in 1948, RAND's mission is to help improve policy and decision-making through research and analysis. One of its critical areas of focus is the development of advanced mathematical models to address complex problems. In the context of disease transmission, RAND has contributed significantly to understanding how diseases spread through networks and populations. By leveraging mathematical formulations and computational simulations in this area, RAND aims to enhance the accuracy of epidemiological models, providing better tools for predicting outbreaks and formulating effective intervention strategies. In the realm of network representations for disease transmission, RAND's goals align with improving public health responses and mitigating the impacts of infectious diseases. They strive to refine the mathematical underpinnings of these models to capture the nuances of real-world interactions more accurately. This includes accounting for various factors such as population density, mobility patterns, and social behavior, which influence the spread of diseases. By developing more sophisticated and representative network models, RAND seeks to provide policymakers and health professionals with the insights needed to design more effective containment and mitigation strategies, ultimately contributing to global health security and resilience.

1.1 Problem Statement

Building on RAND's extensive experience in modeling and simulations, the RAND Corporation sponsored our research group for a nine-week project exploring deep generative approaches to network science for social system simulations. Our first goal, synthetic network generation aims to create artificial networks resembling real-world networks. Network fusion, the second goal, involves combining multiple sub-networks into a single unified network. While these goals have been approached in other network contexts, solutions for large social networks with hundreds of thousands of individuals have not yet been developed. Generating synthetic networks that are similar to those from a training dataset has previously been partially implemented by RAND researchers, but these methods did not take advantage of the speed and accuracy advantages of newer graph generative network models. To the knowledge of the authors, network fusion has not been implemented on large networks. By replacing previous generative networks to create graphs that are more complex and realistic and developing a method to fuse network datasets to enrich social network data, this project helps to improve the ability of social science researchers to understand and utilize complex networks in social simulations.

1.2 Approach

We separated our approach into network generation and network fusion components. The primary objective of network generation is to create synthetic networks that accurately reflect given demographic distributions, such as age and income, while ensuring edge connections and their weights align with target statistics. This involves producing large-scale synthetic networks that are statistically and structurally similar to existing sociocentric networks using a dataset from Portland, Oregon. To achieve this, we explored multiple deep generative methods and compared them with simulated annealing techniques, considering the potential of hybrid approaches. After choosing the best models, we adapted them to work for large-scaled networks with node and edge-features. In order to test the success of the model in generating similar networks, we created a list of statistical graph measures that determine how information passes through a network. These methods, which include degree distribution and clustering coefficients, are our benchmarks for model success.

For network fusion, the goal is to integrate sociocentric datasets with egocentric network data. This involves incorporating additional node features from the egocentric data into the synthetic networks while maintaining structural equivalence with the sociocentric data. Additionally, the project aimed to develop methods for generating population and network models for entirely new cities which we tackled using our fusing approach. Our approach for network fusion includes analyzing egocentric networks to predict individual behaviors based on demographics and social connections, generating joint demographic and behavioral matrices for both datasets, and applying regression-based methods. When these regression-based methods are applied to the sociocentric dataset, values for additional node features can be estimated, enriching the sociocentric dataset.

1.3 Report Overview

This report is structured as follows: Chapter 2 focuses on data analysis, with an emphasis on sociocentric and egocentric graphs, which pertain to entire communities and individual actors, respectively. In this section, we discuss the datasets employed and our methodologies for generating synthetic node and edge features. Chapter 3 explores the challenges of generative modeling, detailing the refined models developed for this task and their potential application in other network generation contexts. Chapter 4 addresses the complexities of network dataset fusion, providing an in-depth discussion of the models we developed and their broader applicability. Chapter 5 presents the experimental results, while Chapter 6 concludes the document by summarizing the key findings and contributions of our work.

Chapter 2

Data Analysis

We focus on two specific types of network datasets: sociocentric and egocentric networks. A network, or graph, consists of nodes (entities) and edges (relationships between entities). A sociocentric network represents an entire population, with each node as a research subject and all ties between subjects included. These networks are broad, capturing connections across the whole population, but shallow, typically containing only a few key demographic attributes and limited edge information. In contrast, egocentric networks, or personal networks, center around an individual (the ego) and include only their immediate connections (alters). These networks are narrow, focusing on the ego's personal community, but deep, often containing detailed attributes about the nodes.

2.1 Sociocentric Data

Sociocentric network data represents interactions within an entire community, providing a comprehensive view of social structures and connections. In this study, we utilize a specific sociocentric dataset that focuses on a synthetic population of Portland, Oregon. Developed by the Network Dynamics and Simulation Science Laboratory (NDSSL) at the Biocomplexity Institute of Virginia Tech, this dataset captures daily social contact networks among individuals within the city. The NDSSL network dataset, created as part of the TRANSIMS project through agent-based modeling, offers a detailed snapshot of daily interactions, including geographic information about the location, duration, and purpose of these interactions.

The dataset includes demographic attributes at both individual and household levels, such as age, gender, household income, and household location. Approximately 1.7 million individuals are organized into 630,000 households, documenting about 20 million distinct interactions. Figure 2.1 represents a subset of the NDSSL network. Each interaction is associated with a duration and set of activities. The data is structured into node-level data, containing individual and household-level information, and edge-level data, describing interactions and associated activities. Edge weights in these graphs reflect cumulative interaction durations. This synthetic dataset is used to inform transmission dynamics models for infectious diseases, evaluate public health interventions, and simulate transportation and social behavior patterns. It is particularly valuable because it offers a realistic representation of daily social interactions, capturing the complexity and diversity of human behaviors within the city while being free from privacy concerns.

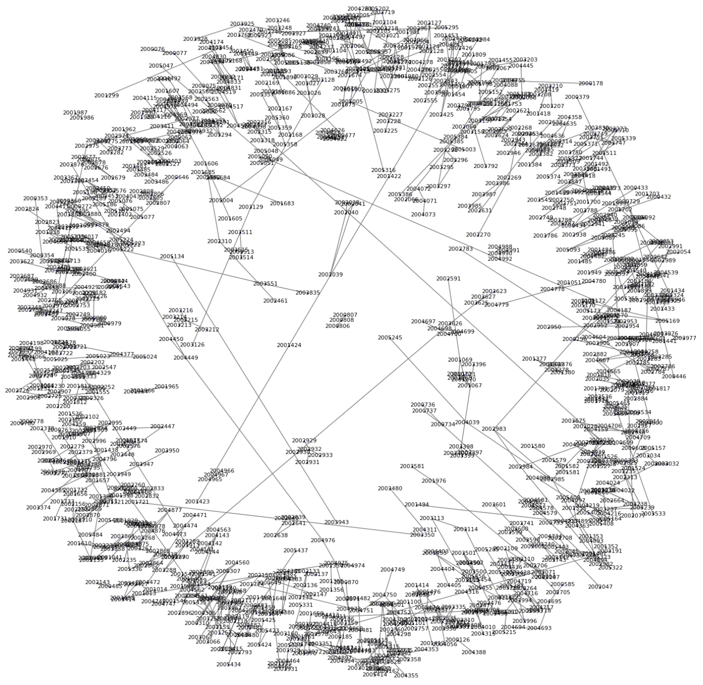


Figure 2.1: A 1000 node subset of the 1.6 million node sociocentric dataset

2.2 Egocentric Data

Egocentric datasets focus on the personal social networks of individual respondents. Each egocentric network centers around a specific individual, known as the 'ego', and includes their immediate social contacts, referred to as 'alters', as well as the connections between those alters. These networks are subsets of a larger, often unobserved, sociocentric network, providing a localized view of social interactions. The FluPaths dataset, collected through a four-year longitudinal survey by the RAND American Life Panel (ALP) from fall 2016 to spring 2020, is an example of an egocentric dataset. Designed to inform agent-based simulation models of influenza vaccination behavior, the dataset includes detailed information on respondents' social networks, vaccination intentions, risk perceptions, and interactions with healthcare professionals.

Respondents were asked a series of pre- and post-flu season questions over eight waves, capturing data on their immediate social contacts (alters) and connections among alters. The first wave of network data collection, conducted using the EgoWeb tool, involved respondents identifying up to 15 individuals they regularly contact and providing demographic and relational data about these alters, pictured in Figure 2.2. Subsequent waves allowed respondents to update this information. The FluPaths dataset comprises 1,898 respondents (egos) and 19,250 alters, with a total of 82,300 edges, including 62,050 alter-alter ties. Each interaction includes details such as relationship type (ego-alter or alter-alter ties), strength of ties, and frequency of interactions. This rich dataset enables analysis of network structures and behaviors, offering valuable insights into how personal and social-network experiences affect vaccination decisions and disease transmission dynamics over time. By leveraging sociocentric and egocentric datasets, this research can provide a comprehensive understanding of social networks and their impact on health behaviors and outcomes.

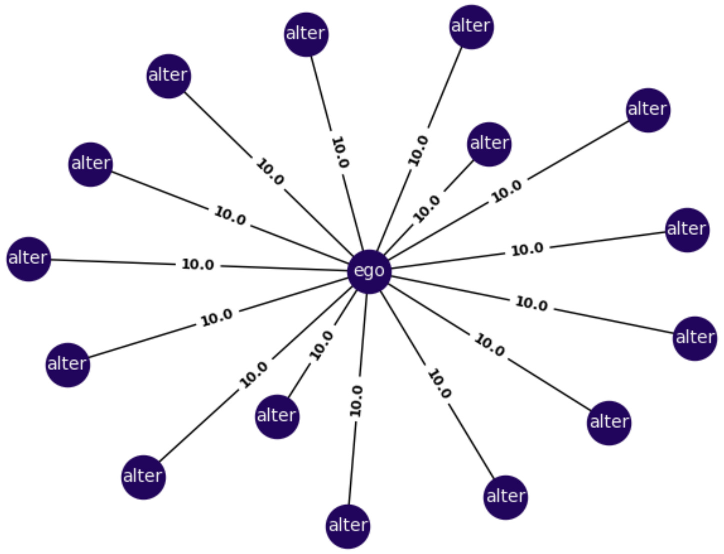


Figure 2.2: An egocentric data point where the 'ego' is pictured in the center with links to their alters

2.3 Data Batching

Given the scale of the sociocentric NDSSL dataset, sampling connected subgraphs is crucial to make the dataset applicable for downstream applications, such as model training.

We employed a breadth-first search (BFS) algorithm to systematically sample these subgraphs. This methodology is analogous to designating a node in NDSSL as the center (ego) and iteratively integrating its neighbors (alters), its neighbors' neighbors, etc. in order to expand the subgraph outward, layer by layer. The sampled subgraphs are further categorized based on their node count into four distinct groups: sub_30 (which contains 10 to 30 nodes), sub_100 (which contains 31 to 100 nodes), sub_300 (which contains 101 to 300 nodes), and sub_1000 (which contains 301 to 1000 nodes). Nodes within the same subgraph share the same graphID, facilitating organized and efficient data handling. We merged the edge features from the NDSSL dataset with the node features from the demographic data to form a cohesive dataset. These enriched subgraphs are stored in CSV format, preserving the integrity of the data for subsequent processing. Moreover, feature engineering techniques are applied to refine the data, which include mapping purpose IDs from strings to discrete numbers and rescaling contact durations to a normalized range between 0 and 1.

In the subsequent phase, the CSV-formatted subgraphs are transformed into graph objects using NetworkX, which is a popular Python library specialized in constructing, manipulating, and analyzing complex networks. Following the conversion, the NetworkX graphs are partitioned into training, validation, and testing sets. Each set is comprised of batches containing a specific mix of subgraphs: 11 sub_30 graphs, 3 sub_100 graphs, 1 sub_300 graph, and 1 sub_1000 graph per batch.

To facilitate future accessibility and utilization, the processed datasets are serialized into pickle files. This serialization process, known as pickling, converts the Python object hierarchy into a byte stream, allowing for efficient storage and retrieval for downstream applications.

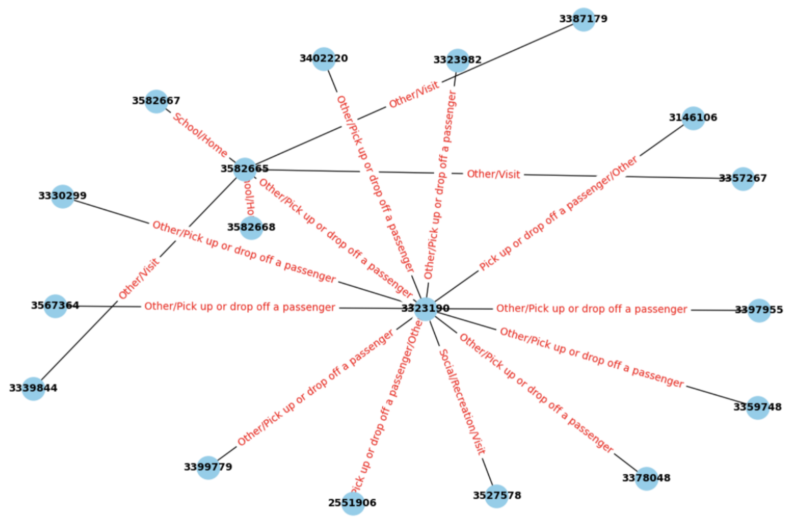

3387179
3402220
3323982
Other/Visit
3582667
School/Home
3146106
3582665
Other/Visit
3357267
3330299
ool/Ho
3582668
Other/Pick up or drop off a passenger
Other/Pick up or drop off a passenger
Other/Pick up or drop off a passenger
Other/Pick up or drop off a passenger
Pick up or drop off a passenger/Other
Other/Visit
3567364
Other/Pick up or drop off a passenger
3323190
Other/Pick up or drop off a passenger
3397955
Other/Pick up or drop off a passenger
3339844
3359748
Other/Pick up or drop off a passenger
Pick up or drop off a passenger/Othe
Social/Recreation/Visit
Other/Pick up or drop off a passenger
3399779
3378048
2551906
3527578

Figure 2.3: A subgraph sampled from the NDSSL dataset during the data batching process

Chapter 3

Generative Modeling

3.1 What are Generative Models?

Generative modeling focuses on creating new graphs that replicate the structural properties of existing networks. This involves learning the underlying distribution of graphs from a given dataset and using that knowledge to generate synthetic networks that maintain similar characteristics, such as node degrees, clustering coefficients, and community structures. Models like GraphRNN [12], Variational Autoencoders (VAEs) [5] for graphs, and Graph Generative Adversarial Networks (GraphGANs) [2, 11] have been developed to handle this task, leveraging deep learning techniques to capture complex dependencies between nodes and edges. One of the primary applications of generative modeling in network generation is in simulating realistic network scenarios for research and practical applications. For example, in social network analysis, generative models can create synthetic social networks that preserve the statistical properties of real-world social interactions, which can be used for testing hypotheses or evaluating new algorithms. In addition, generative models are capable of creating smaller, yet structurally equivalent networks that serve practical purposes, especially when computational resources are limited. By learning the underlying patterns and distributions of large, complex networks, these models can generate reduced versions that retain essential characteristics such as clustering coefficients and degree distributions. This capability is particularly valuable for simulations and analyses that require multiple iterations or are computationally intensive. For instance, in network security, smaller synthetic networks can be used to test intrusion detection systems or simulate attack scenarios without the need for full-scale networks. In research, these smaller networks allow for rapid prototyping and hypothesis testing, making it easier to explore network behaviors and dynamics.

3.2 Challenges of Implementation

Graph Neural Networks (GNNs) [10] are a class of neural networks designed to handle graph-structured data, which is typically represented as $G = (\mathcal{V}, \mathcal{E})$ with $\mathcal{V}$ denoting vertices (nodes) and $\mathcal{E}$ denoting edges. GNNs understand the local and global information about the graphs through node and edge embeddings, which are low-dimensional vectors capturing information about nodes and edges respectively. GNNs leverage an aggregation mechanism known as message passing, in which each node updates its own representation by aggregating information from adjacent nodes, to capture relationships and dependencies through nodes and edges. Despite their powerful capabilities in learning complex graph relationships, GNNs face significant challenges, particularly in computational complexity and memory usage. GNNs require extensive computations due to iterative message passing, which can lead to high computational demands as the graph's size increases. In addition, GNNs consume substantial memory to store nodes features as well as the graph representation, such as large adjacency matrices, during model training stage.

Existing generative models, such as GraphRNN, VAEs for graphs, and GraphGANs have been leveraged to understand graph-structured data. However, due to the challenges of computational complexity and memory usage mentioned above, they are unable to scale beyond a couple hundreds of nodes, limiting their applicability for large-scale network generation for our generative aim. In addition, these models are not node-invariant, meaning their output can vary depending on the order in which nodes are presented. This lack of permutation invariance can lead to inconsistencies and inaccuracies in generated graphs, as the models might not fully capture the true underlying structure and properties of the network.

To generate large-scale synthetic networks, new approaches are needed to address these scalability issues while ensuring node invariance. Developing models that can accurately and efficiently generate large-scale networks while maintaining structural integrity is crucial for advancing the field and enabling practical applications that require realistic synthetic data. As such, we looked into two state-of-the-art models that possess these ideal properties, namely score-based diffusion model and iterative local expansion model.

3.3 Score-Based Generation

A Score-Based Models Overview

The model that we ultimately used to generate realistic graphs utilizes score-based diffusion. Diffusion models [3] are a class of generative models that work by first adding noise to an entity (such as an image or graph) in gradual steps, transforming it into pure noise. This process creates a series of increasingly noisy images. The diffusion model is then trained to reverse this process, learning how to denoise the image step by step. Starting from the noisy image, the model predicts and removes noise incrementally, gradually reconstructing the original image. The intuition is that by understanding how to go from noise to a clear image, the model can generate realistic images or graphs from random noise. Mathematically, the model learns how to transfer the distribution of the noisy image, which is simple, to the more complex distribution, that is the original image or graph.

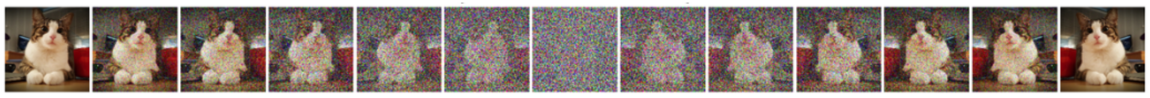


Figure 3.1: Visualization of the Diffusion and Denoising Processes

Score-Based Generative Modeling is a type of diffusion modeling which does not directly find the underlying data distribution. While most generative models seek to determine the distribution of the training data, score-based models only seek to determine gradients of the data, which is easier for the model to learn. Learning the gradients, or area of steepest increase, lets the model learn the most probable transformations from the prior distribution to the data distribution and vice versa. This simplification is especially useful for incredibly complex data, like graphs with millions of nodes.

B Score-Based Model Mathematical Foundation

The score-based generative modeling process begins by gradually adding noise to the data through a stochastic differential equation (SDE), transforming the original data distribution into a prior distribution, typically Gaussian noise. Mathematically, this noising process can be described by the forward SDE in Equation (3.1), where x represents the data, f and g are functions that control the drift and diffusion, respectively, and w is a Brownian motion.

$$dx = f(x,t)\,dt + g(t)\,dw. \tag{3.1}$$

To generate new data, the model learns the score function, $\nabla_x \log p_t(x)$, which is the gradient of the logarithm of the data density at each level of noise. This score function guides the reverse SDE, which aims to denoise the data step by step. The reverse SDE is given by Equation (3.2), where $\nabla_x \log p_t(x)$ is the score function learned during the forward process. Equation (3.2) reverses the forward noising process, transforming noise back into structured data.

$$dx = [f(x,t) - g(t)^2 \nabla_x \log p_t(x)]dt + g(t)dw. \tag{3.2}$$

The use of Brownian motion w in both the forward and reverse SDEs ensures that the process is stochastic, allowing the model to generate diverse samples. By iteratively applying the reverse SDE starting from the prior distribution, the model reconstructs data that resembles the original dataset. This approach combines the theoretical foundations of stochastic differential equations and score matching, enabling the creation of high-quality synthetic data across various domains.

3.4 Iterative Local Expansion Model

A Iterative Local Expansion Model Overview

The iterative local expansion model [1] consists of a forward and backward process. In the forward coarsening process, the graph is compressed step by step until it becomes a single node containing information about the entire graph. In the backward expansion and refinement process, the model will learn how to invert the coarsening process by expanding each node in the compressed graph into a cluster of nodes and recover the appropriate local structure within this newly-expanded local subgraph by adding the intercluster and intracluster edges. The iterative local expansion model outperforms existing GNNs because it retains both global and local structure. Through the gradual expansion and refinement in the backward process, the model builds first the global high-level presentation of the graph and then refines the local details. In addition, the model retains high expressive power in the local subgraphs because it is explicitly trying to capture complex relationships during local expansion. Last but not least, it is able to scale to thousands, instead of hundreds, of nodes because it avoids modeling the joint distribution for all node pairs in the entire graph by performing expansion locally.

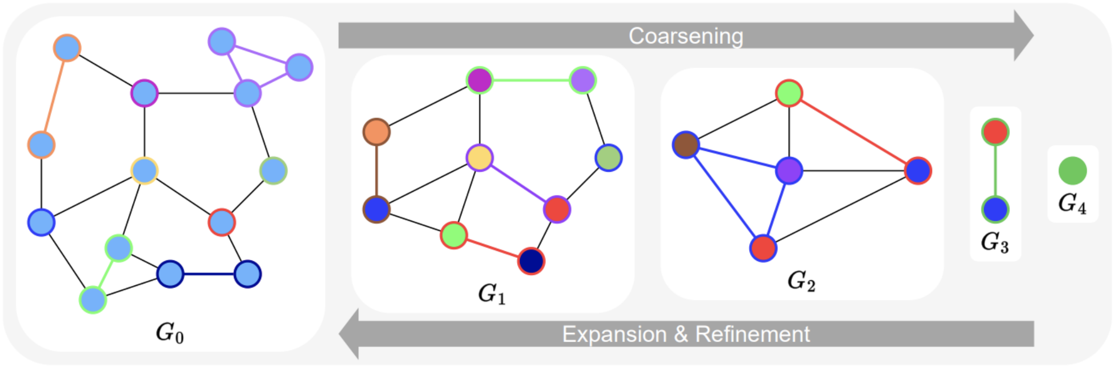


Figure 3.2: High-Level Overview of Iterative Local Expansion Forward and Backward Processes

B Iterative Local Expansion Mathematical Foundation

To formalize each stage in the iterative local expansion model, we define graph coarsening as follows: let G = $(\mathcal{V}, \mathcal{E})$ denote a graph G consisted of node set $\mathcal{V}$ and edge set $\mathcal{E}$. Let $\{\mathcal{V}^{(1)}, \mathcal{V}^{(2)}, ..., \mathcal{V}^{(\overline{n})}\}$ by a partitioning of $\mathcal{V}$ such that $\forall \mathcal{V}^{(i)} \in \mathcal{V}$, $\mathcal{V}^{(i)}$ gets compressed into a single node $\overline{v}^{(i)} \in \overline{V}$, where $\overline{V}$ denotes new node set after compression. The edge $e^{(a,b)} \in \overline{E}$, which is the edge set after compression, if and only if there exists an edge e connecting $v^{(i)} \in \mathcal{V}^{(a)}$ and $v^{(j)} \in \mathcal{V}^{(b)}$ and $\mathcal{V}^{(a)} \cap \mathcal{V}^{(b)} = \emptyset$.

Graph expansion can be defined as: let G = $(\mathcal{V}, \mathcal{E})$ denote a compressed graph G consisting of node set $\mathcal{V}$ and edge set $\mathcal{E}$. Given a cluster size vector $\mathbf{v} \in \mathbb{N}^n$ (where $n = |\mathcal{V}|$) denoting the expansion size for each node $\mathcal{V}^{(i)}$ in $\mathcal{V}$, $\forall \mathcal{V}^{(i)} \in \mathcal{V}$, $\mathcal{V}^{(i)}$ gets expanded into a cluster of $f(\mathcal{V}^{(i)}, \mathbf{v}_i)$ nodes denoted by $\tilde{V}^i$, where f maps a node $\mathcal{V}^{(i)}$ in the compressed graph to $|\tilde{V}^{(i)}| \in \mathbb{N}$. New intercluster and intracluster edges are added to the graph during the expansion, each with a probability $0 \leq p \leq 1$, for all distinct nodes $v^a, v^b \in \tilde{V}$ whose distace is less than a radius $r \in \mathbb{N}$ to form a new edge set $\tilde{E}$. This perturbed expansion is denoted as $\tilde{G}(G, \mathbf{v}) = (\tilde{\mathcal{V}}, \tilde{\mathcal{E}})$.

Graph refinement takes in the perturbed expansion $\tilde{G} = (\tilde{\mathcal{V}}, \tilde{\mathcal{E}})$ and an edge selection vector $\mathbf{e} \in \{0, 1\}^{|\tilde{\mathcal{E}}|}$. It will selectively eliminate the ith edge from $\tilde{G}$ to obtain the refinement of $\tilde{G}$ if and only if $\mathbf{e}_i = 0$. This refinement is denoted by $G(\tilde{G}, \mathbf{e}) = (\mathcal{V}, \mathcal{E})$.

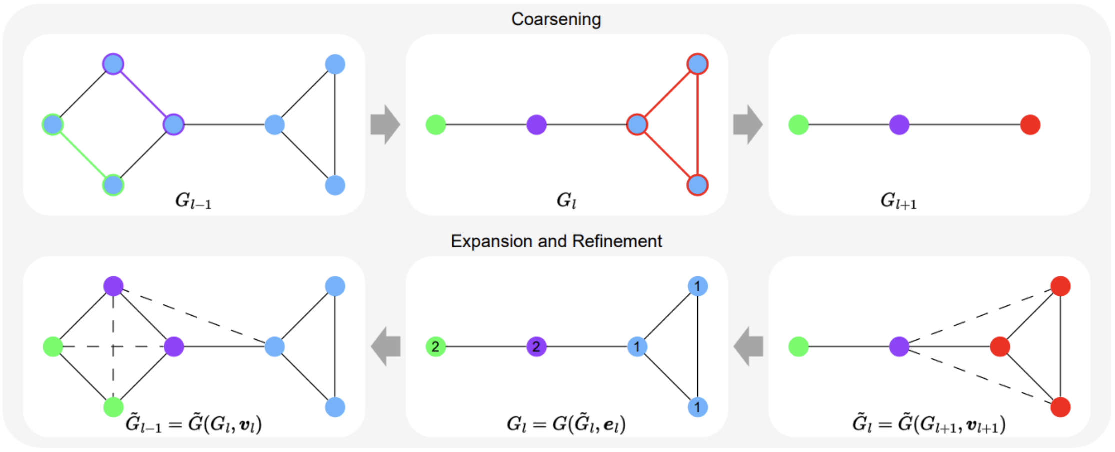

Coarsening
G_{l-1}
G_l
G_{l+1}
Expansion and Refinement
$\tilde{G}_{l-1} = \tilde{G}(G_l, \boldsymbol{v}_l)$
2
2
1
1
1
$G_l = G(\tilde{G}_l, \boldsymbol{e}_l)$
$\tilde{G}_l = \tilde{G}(G_{l+1}, \boldsymbol{v}_{l+1})$

Figure 3.3: Detailed Breakdown of Iterative Local Expansion Forward and Backward Processes From Timestep $l - 1$ to $l + 1$

C Iterative Local Expansion Implementation Details

The Python implementation of the iterative local expansion algorithm for graph generation leverages several key packages and machine learning methods to achieve accurate and efficient results. The primary packages used include NetworkX for graph manipulation, Pandas for data handling, and Numpy for numerical computations. NetworkX is essential for creating and expanding the graph structures, offering a robust framework for graph operations such as node and edge addition, neighborhood exploration, and graph refinement. Pandas and Numpy are utilized for efficiently managing the large datasets and performing the necessary numerical calculations, including matrix operations and statistical analysis, which are crucial for refining the graph structure during the iterative expansion process.

In addition to these foundational packages, more advanced machine learning methods are incorporated to guide the expansion and ensure the generated graph retains the desired properties of the target graph. Techniques such as node embedding and graph similarity measures are applied to assess and refine the graph at each iteration. Node embeddings, which transform the nodes into a continuous vector space, help in capturing the complex relationships between nodes and guide the selection of new nodes to add during expansion. Graph similarity measures, on the other hand, are used to compare the generated graph with the target graph, ensuring that the local and global structural properties are preserved throughout the iterative process.

The validation of the generated graphs is conducted using a variety of metrics from the GraphGym (gg) library, which provides comprehensive tools for evaluating the structural fidelity and uniqueness of the generated graphs. The key validation metrics include:

- `NodeNumDiff()`: This metric compares the difference in the number of nodes between the generated and target graphs, ensuring that the graph expansion is occurring at an appropriate rate.
- `NodeDegree()`: Evaluates the distribution of node degrees, ensuring that the generated graph maintains a similar connectivity pattern to the target graph.
- `ClusteringCoefficient()`: Measures the tendency of nodes to form tightly-knit groups, a critical aspect of ensuring the generated graph mirrors the clustering properties of the target graph.
- `OrbitCount()`: Examines the local graph structures by counting graph orbits, providing insight into the similarity of local substructures between the generated and target graphs.
- `Spectral()`: Analyzes the eigenvalues of the graph's adjacency matrix, offering a view of the global structural properties, such as community structure and graph connectivity.
- `Wavelet()`: Utilizes wavelet transforms to capture multi-scale information about the graph, ensuring that both local and global patterns are preserved in the generated graph.
- `Ratio()`: This metric compares various ratios, such as edge-to-node ratios, to assess the proportionality and balance of the graph structure.
- `Uniqueness()`: Evaluates how distinct the generated graph is compared to previously generated graphs, ensuring that the algorithm produces diverse graph structures rather than repetitive ones.

- `Novelty()`: Measures the introduction of new structural patterns that are not present in the target graph, which is important for generating innovative graph structures.

These metrics collectively ensure that the generated graphs are not only similar to the target graphs in terms of basic structural properties but also exhibit desirable characteristics such as diversity, novelty, and structural fidelity, making the iterative local expansion algorithm a powerful tool for generating complex and realistic graph structures.

3.5 Generative Results

In this section, we present the results of our generative graph model. As illustrated in Figure 3.5, the model demonstrates a strong ability to generate graphs with intricate structures, effectively capturing and modeling the interactions between various nodes. Despite encountering limitations in processing power, which restricted our ability to generate graphs with more than 300 nodes, the model performed well within this constraint. The generated graphs exhibit a high degree of structural complexity, underscoring the model's capability to replicate the nuanced interconnections typical of larger networks. These results are promising and suggest that, with enhanced computational resources, the model could scale to even larger graphs, potentially unlocking deeper insights into the complex relationships within more extensive networks.

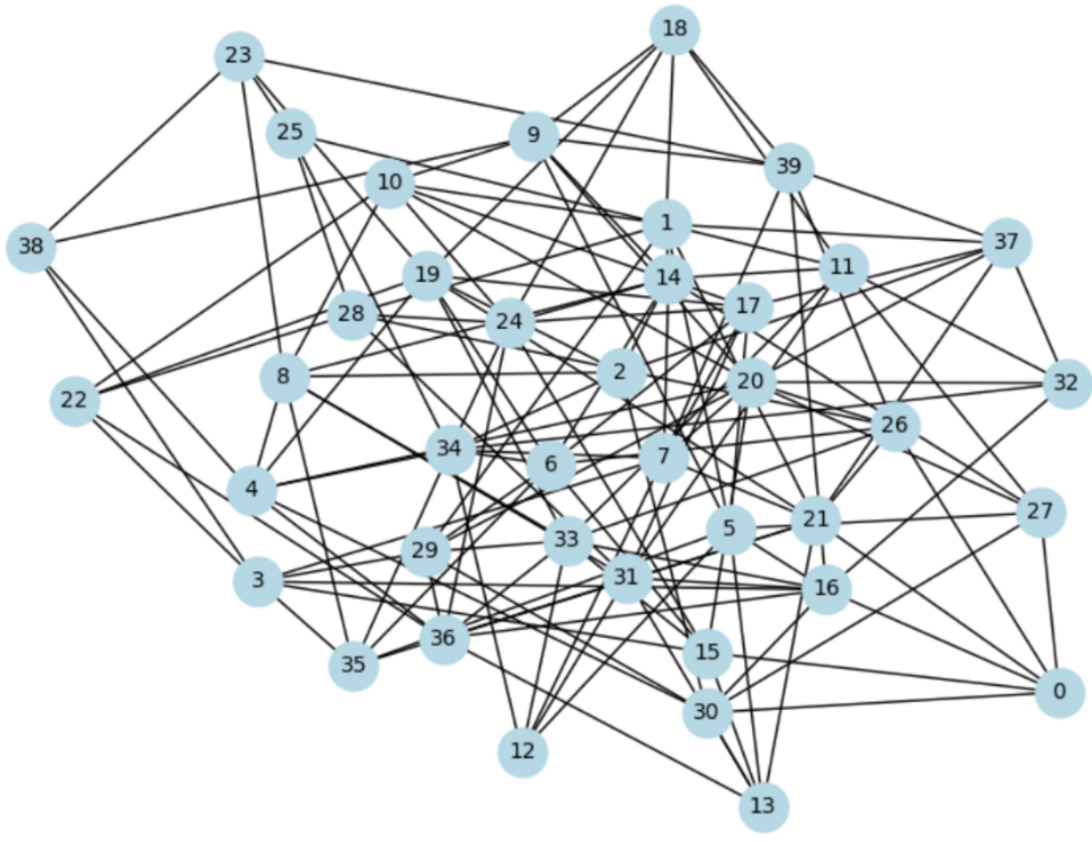


Figure 3.4: An Example of a Graph Generated by Iterative Local Expansion Model

Chapter 4

Network Fusion

4.1 What is Network Fusion?

Network fusion aims to combine multiple sub-networks into a single unified network, which is crucial for integrating information from various sources to make network data more comprehensive. To our knowledge, network fusion has not yet been implemented on large graphs, such as those from the NDSSL and FluPaths data, theoretical approaches have been described.

The motivation for fusing egocentric and sociocentric networks lies in the potential to enhance the representation and analysis of social systems. By combining the detailed personal interactions captured in egocentric networks with the broader structure of sociocentric networks, this approach enables more accurate modeling of data spread and information diffusion. This integration leverages diverse data sources to improve the quality and depth of network analysis, supporting advanced simulations that can inform policy decisions, public health strategies, marketing campaigns, urban planning, and emergency response efforts. The primary objective of this project is to fuse egocentric networks with a sociocentric network to enhance the representation and analysis of social systems. Specifically, the project aims to integrate node features from egocentric networks into a comprehensive sociocentric dataset from Portland. This integration will enable the modeling of data spread and information diffusion within the broader social network, leveraging the detailed personal interactions captured in egocentric networks. Fully mapping a large-scale network for a population in real-world scenarios is often difficult or impossible. Policy researchers frequently encounter multiple, partially overlapping datasets describing the population of interest. Just as data fusion integrates multiple data sources, network fusion tackles the challenge of merging data with a network structure. Network fusion involves creating a new, large-scale network that combines aspects of two distinct network datasets.

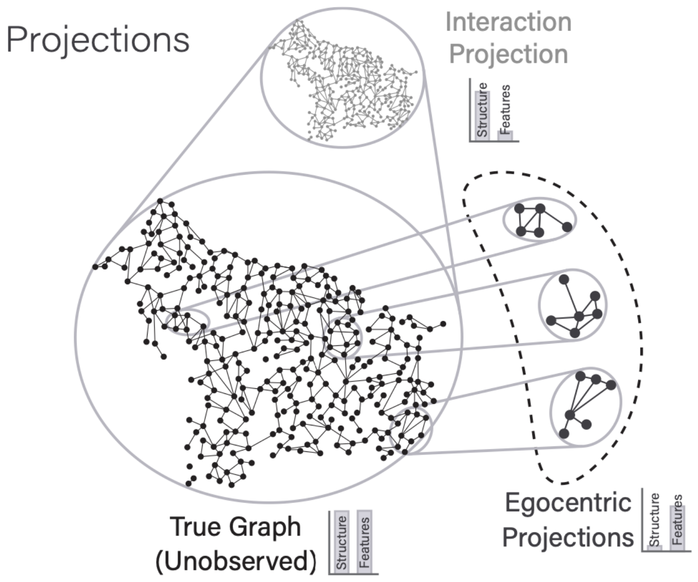


Figure 4.1: Illustration of network data fusion. We postulate that there exist a true unobserved graph that is consistent with our NDSSL sociocentric network data (i.e., the interaction projection) and the FluPaths ALP egocentric data (i.e., egocentric projections).

4.2 Challenges of Implementation

Network fusion, or graph merging, faces significant limitations when dealing with large-scale networks, particularly in terms of scalability and structure retention. Scalability issues arise because as the size of the graphs increases, the computational complexity of merging them grows exponentially. This is due to the need to align and integrate nodes and edges from multiple large networks, which can lead to prohibitive memory and processing requirements. Additionally, preserving the structural integrity of the original networks during fusion becomes challenging. As networks grow, maintaining accurate node and edge relationships while integrating different datasets becomes increasingly complex, risking the loss of crucial structural details and relationships.

Algorithms like Supreme [4] and Affinity Network Fusion [7], though effective for smaller networks, struggle with larger graphs due to their design limitations. Supreme, which relies on optimizing a global objective function for merging, can become computationally infeasible as network size increases. The need to handle a vast number of potential alignments and interactions makes the optimization process slow and resource-intensive. Affinity Network Fusion, on the other hand, focuses on similarity measures between nodes and edges to guide merging. While it works well for smaller networks, its reliance on pairwise affinity calculations does not scale effectively, as the number of potential comparisons grows rapidly with network size, leading to performance bottlenecks and reduced accuracy in capturing large-scale network dynamics.

A The GRAF Model

GRAF [8] is a computational approach to network fusion operating graph convolutions on multiple graph-structured data with the utilization of attention mechanisms and network fusion simultaneously. Briefly, the first step is data collection and neighborhood generation. In the second step, we obtain node-level and association-level attention. In the third step, GRAF fuses multiple networks into one weighted network utilizing node-level and association-level attention mechanisms. In the last step, GRAF learns the node embeddings using GCN with the convolutions on the fused network.

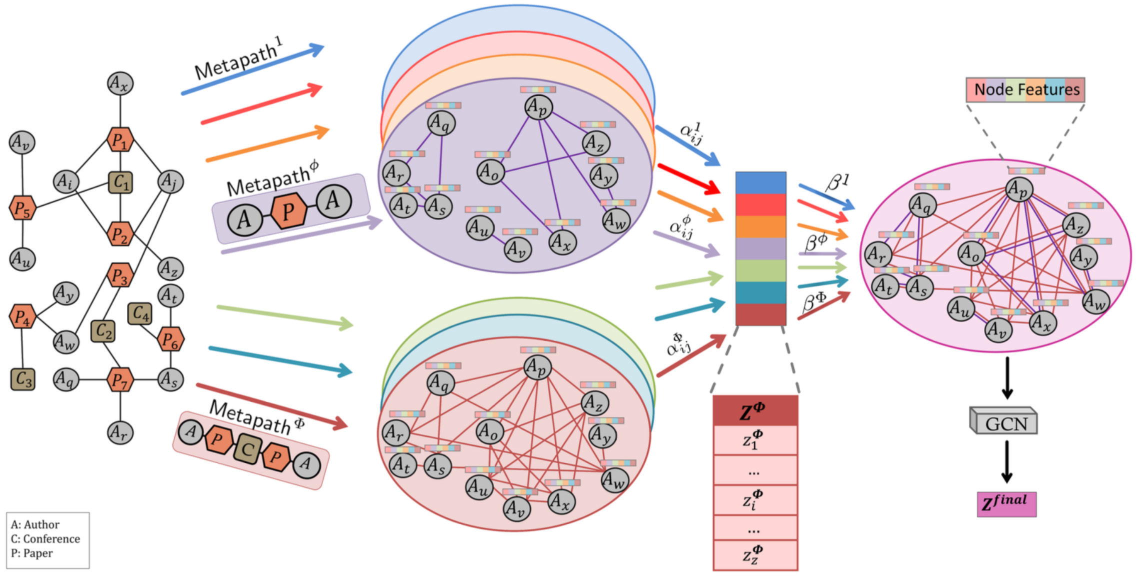


Figure 4.2: GRAF pipeline on a heterogeneous network.

GRAF is a computational approach that applies graph convolutions to multiple graph-structured datasets, using attention mechanisms and network fusion simultaneously. The process involves four main steps:

1. Data Collection and Neighborhood Generation: Initially, GRAF generates neighborhoods based on meta-paths for heterogeneous networks or similarities for homogeneous networks, forming a meta-path-based network.

2. Node-Level and Association-Level Attention: GRAF then calculates the importance of each neighbor per association (node-level attention) and the significance of each association for the prediction task (association-level attention). This is achieved using a hierarchical attention network (HAN) architecture.

3. Attention-Aware Network Fusion: Multiple networks are fused into one weighted network by combining node-level and association-level attention. This step prioritizes edges based on their importance and association weights, creating a comprehensive and weighted fused network.

4. Node Embedding Learning: Finally, GRAF learns node embeddings using a Graph Convolutional Network (GCN) on the fused network. The process includes edge elimination to remove weak edges and optimize the network structure. Node embeddings are generated and used for downstream tasks like node classification.

The motivation behind GRAF is to utilize attention mechanisms to weigh node associations and integrate multiple networks into a single, cohesive structure, improving the accuracy of graph convolutional operations and the overall performance of predictive tasks. This method is particularly useful for handling complex, heterogeneous networks by capturing both local node-level and global association-level information. Although we ultimately decided not to implement this method, a similarly complex method capable of creating entirely new graphs with altered structures is an important next goal for network fusion.

4.3 Network Fusion Model

A Fusion Algorithm Overview

The ultimate model for network fusion is entirely statistical and utilizes the conditional probabilities of vaccination in the egocentric network to estimate the probabilities of vaccination in the sociocentric network. Then an iteration scheme is applied to the initial probabilities of vaccination in the sociocentric network to simulate the effect that interactions with others have on an individual's probability of vaccinating. This algorithm does not edit the structure of the sociocentric algorithm. Our proposed algorithm is as follows:

1. Extract demographic statistics such as age and gender from the egocentric and sociocentric datasets.

2. Generate joint demographic mixing matrices for both the generated sociocentric and egocentric network data. This involves examining matrices representing age group, gender, and household member or non-member mixing patterns within both networks. Egocentric mixing matrices contain the probability distribution information depending on the demographics of each individual.

3. Assign an initial probability of vaccination $p_i(0)$ from the egocentric mixing matrix conditioned on the demographic data for each node in the sociocentric network.

4. Utilize an iterative formula (4.1) to calculate vaccination probabilities at each time step until the desired convergence is achieved.

$$p_i(t+1) = \alpha p_i(t) + (1-\alpha)\frac{\sum_{j\in B(i)} w_{ij}\cdot g(p_j(t))}{\sum_{j\in B(i)} w_{ij}},\ i,j \in [1,..,N] \tag{4.1}$$

Given the sociocentric graph $G = (\mathcal{V}, \mathcal{E})$ with $\mathcal{V}$, where $\mathcal{V}$ is the set of vertices, E is the set of edges and number of nodes $N = |\mathcal{V}|$. α is a parameter $(0 < \alpha < 1)$ representing the weight given to the individual's current belief versus the influence from neighbors.

$$p_i(t) = \mathbb{P} \text{ of vaccination at time t}$$

$$w_{ij} = e_{ij} \text{ edge weight} \in [0,1]$$

$$B(i) = \{j \in \{1,...,N\} | \exists e_{ij} = (v_i, v_j) \in \mathcal{E}\}$$

$$g : [0,1] \to \mathbb{R} \text{ continuous and bounded function.}$$

The update rule (4.1) models changes in vaccination probabilities and node attributes by taking into account social interactions between individuals. It aggregates the influence of all neighbors and updates the vaccination probability of the individual. The edge weight represents the proportion of time that two individuals interact, and the term $\frac{\sum_{j\in B(i)} w_{ij}\cdot g(p_j(t))}{\sum_{j\in B(i)} w_{ij}}$ normalizes the influence of neighbors.

The model converges when the vaccination probabilities $p_i(t)$ for all nodes i stabilize, meaning they no longer change significantly with further iterations $p_i(t+1) \approx p_i(t)$ for $\forall i \in [1, .., N]$.

$$\text{Stopping criterion: } \max_{v_i \in V} |p_i(t+1) - p_i(t)| \le \epsilon$$

For the iteration model, an equilibrium point p_i^* satisfies:

$$p_i^* = \alpha p_i^* + (1-\alpha) \frac{\sum_{j \in B(i)} w_{ij} \cdot g(p_j^*)}{\sum_{j \in B(i)} w_{ij}} \tag{4.2}$$

$$p_i^* = \frac{\sum_{j \in B(i)} w_{ij} \cdot g(p_j^*)}{\sum_{j \in B(i)} w_{ij}} \tag{4.3}$$

We expect the model to converge towards the equilibrium.

The algorithm proposed is based on data statistics rather than the network structure, which is advantageous for our model. Egocentric data may not always be structurally representative, as it depends on individuals' responses to the FluPath survey. Now, let us delve deeper into the structure of the egocentric network:

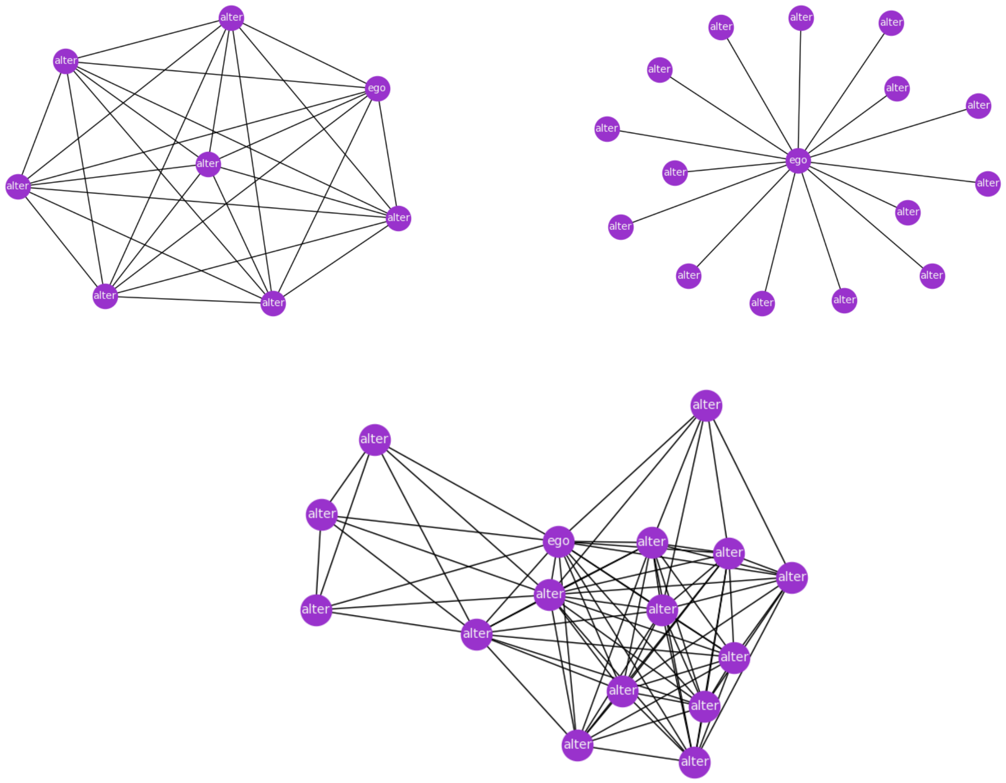


Figure 4.3: Examples of egocentric network structures from the FluPaths dataset.

In Figure 4.3, various egocentric network structures are showcased from the FluPaths dataset. The initial graph demonstrates a fully connected network, which is sub-optimal for model training. Conversely, the second graph lacks information about edges between alters. The third graph provides a comprehensive representation of the egocentric network, displaying a diverse range of ego-alter and alter-alter connections with varying strengths.

Our approach is focused on the mixing matrices instead. Mixing matrices are crucial in network fusion as they capture the interaction patterns between different node categories, helping to preserve the structural integrity and meaningful relationships within the fused network. They ensure consistency across networks, guide the fusion process, and enhance the interpretability of the resulting network by maintaining key interaction dynamics from the original networks. Additionally, mixing matrices help ensure statistical equivalence between the original and fused networks, allowing the fused network to accurately reflect the properties and behaviors of the individual networks. We generated demographic mixing matrices. In Figure 4.4, an example of the age mixing matrix sampled from the sociocentric dataset can be found below.

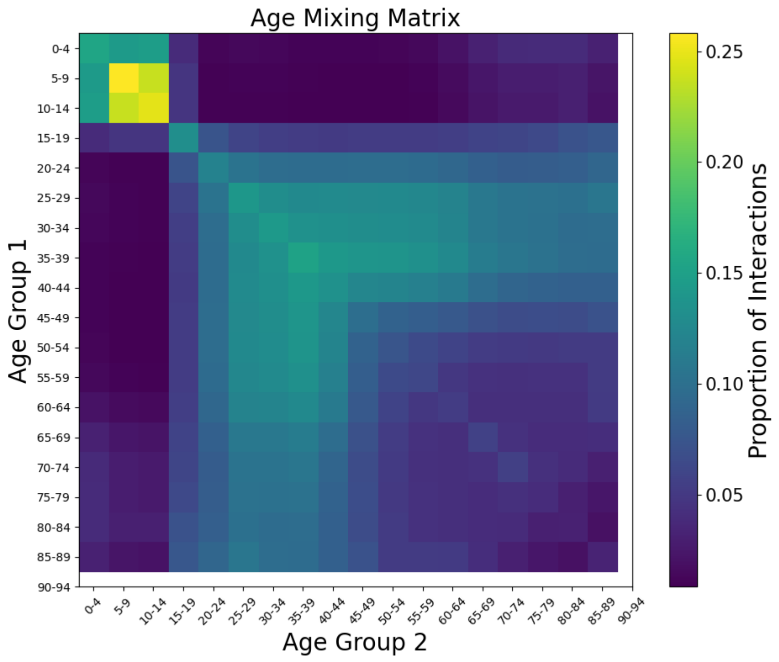


Figure 4.4: Age mixing matrix - NDSSL data

B Fusion Implementation Details

The Python implementation of the network fusion algorithm begins with the construction of a 6x2x2 mixing matrix that represents the distribution of individuals within an egocentric network based on age, gender, and vaccination status. The matrix is populated by filtering and categorizing the network data into six age groups, two gender categories, and two vaccination statuses. For each combination of age and gender, the number of vaccinated and unvaccinated individuals is counted and placed into the corresponding indices of the matrix. This provides a structured representation of the egocentric network's demographic and vaccination composition.

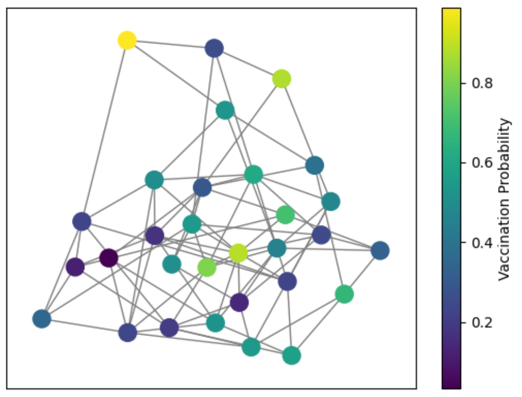
Next, the initial vaccination probabilities are derived from the mixing matrix by calculating the proportion of vaccinated individuals within each age-gender group. These probabilities serve as the baseline estimates for the vaccination status of individuals in a larger sociocentric network. Each node in the sociocentric network corresponds to an individual characterized by age and gender attributes. The initial vaccination probability for each person in the sociocentric network is assigned based on the corresponding age and gender group in the egocentric mixing matrix.

To refine these initial estimates, the iterative updating scheme is applied. In each iteration, the vaccination probability of each node is adjusted based on the vaccination probabilities of its neighbors in the sociocentric network. This process accounts for the influence of social connections, with the idea that individuals are more likely to be vaccinated if their neighbors are also vaccinated and vice versa. The iteration continues until the vaccination probabilities converge or until a predefined maximum number of iterations is reached.

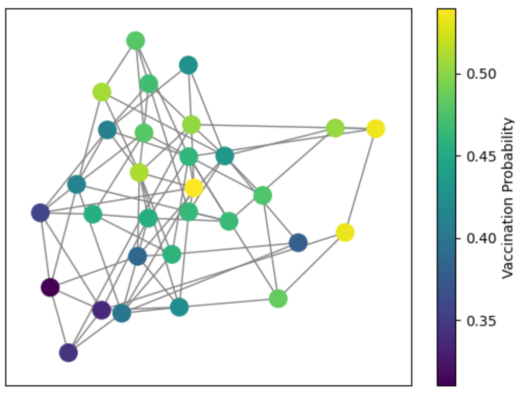
This iterative approach, combined with the initial probabilities from the egocentric network, enables a more accurate estimation of vaccination probabilities across the sociocentric network, reflecting both individual demographic factors and the influence of social ties. The use of Pandas and Numpy facilitated the data handling and numerical operations required for constructing the mixing matrix and updating probabilities, while NetworkX provided the necessary tools for modeling and iterating through the sociocentric network.

4.4 Fusion Results

Ultimately, we were able to estimate the vaccination probabilities, which successfully converged. The algorithm demonstrated convergence across multiple trials, with the vaccination probabilities stabilizing to equilibrium values after a finite number of iterations. The convergence rate was influenced by the choice of the parameter α, the influence function $g(\cdot)$, and the network structure. The weights w_{ij} representing communication frequency, played a significant role in shaping the dynamics. Nodes that communicated more frequently with highly influential neighbors tended to converge more rapidly towards their neighbors' vaccination probabilities. A simulation example on a small network is provided below to illustrate the dynamics of the model. One method of verification we conceptualized was generating mixing matrices for the egocentric graph and sociocentric graphs and comparing them based on the conditional probabilities of vaccination for different attribute groups. However, developing and implementing validation metrics is a next step.



(a) Graph before fusion



(b) Graph after fusion

Figure 4.5: Network fusion example on graph.

Figure (a) 4.4 illustrates a sampled graph with 30 nodes that already contains the initial probability of vaccination summed from the egocentric dataset. The function $g(p_i(t)) =$

$p_i(t)$ was chosen to be identity function, $\alpha = 0.5$ and $\epsilon = 0.01$. The second graph (b) 4.4 illustrates the network after iteration process and we can observe how the probabilities change towards the equilibrium of the system.

Chapter 5

Conclusion

Our research aims to advance microsimulation and agent-based models by integrating sociocentric and egocentric network datasets to create more detailed and accurate simulations of social interactions. The proposed approach involves generating synthetic networks using the iterative expansion model and performing network fusion to enhance these networks with additional individual-level features.

To achieve this, we started by batching the sociocentric network data using Breadth First Search and creating subgraph datasets of different sizes to train the model on. This process is designed to produce smaller, statistically similar subgraphs, addressing potential overfitting issues and improving dataset manageability. The next step involved applying the iterative expansion model for graph generation. This model refines network details by compressing the graph into a single node and subsequently expanding it to recover local structures. This approach aims to preserve both global and local details of the network, thereby ensuring that the synthetic networks are statistically and structurally analogous to the original sociocentric datasets.

Following this, the focus shifted to network fusion, which entailed integrating sociocentric and egocentric data to create enriched synthetic networks. This phase involved combining diverse data sources while maintaining structural integrity, a task that presents challenges due to the limited existing literature on such data integration. Addressing these challenges will be crucial to advancing the field. Future work will concentrate on developing methods for generating network data specific to new cities, which will require learning how to change the network structure of sociocentric graphs. Future models will need to capture the unique characteristics of different individuals and how their opinions change upon interacting with other people, furthering our understanding of social systems.

Overall, this research is designed to enhance the accuracy and detail of social simulations through innovative techniques in graph generation and network fusion. By tackling the challenges inherent in these processes, the research promises to offer valuable insights into human behavior and improve decision-making processes in social simulations.

Chapter 6

Abbreviations

ALP. American Life Panel. A RAND Corporation internet panel used to study U.S. public opinion, behavior, and policy impacts.

BFS. Breadth-First Search. An algorithm for traversing or searching tree or graph data structures, exploring all neighbors at the present depth before moving on to nodes at the next depth level.

GNN. Graph Neural Network. A type of neural network designed to process data represented as graphs, capturing dependencies between nodes via message passing.

GraphGAN. Graph Generative Adversarial Network. A model that combines graph structures with GANs to generate realistic graph data by training a generator and a discriminator in competition.

GraphRNN. Graph Recurrent Neural Network. A neural network model designed for generating graphs sequentially, node by node, or edge by edge, recurrently.

IPAM. Institute for Pure and Applied Mathematics. An institute of the National Science Foundation, located at UCLA.

RIPS. Research in Industrial Projects for Students. A regular summer program at IPAM, in which teams of undergraduate (or fresh graduate) students participate in sponsored team research projects.

SDE. Stochastic Differential Equation. An equation that models systems influenced by random noise, commonly used in fields like physics, finance, and biology.

UCLA. The University of California at Los Angeles.

NDSSL. Network Dynamics and Simulation Science Laboratory.

VAE. Variational Autoencoder. A type of generative model that learns to encode input data into a latent space and then decode it back, often used for generating new data similar to the training set.

Selected Bibliography Including Cited Works

[1] A. Bergmeister, K. Martinkus, N. Perraudin, and R. Wattenhofer, *Efficient and scalable graph generation through iterative local expansion*, in The Twelfth International Conference on Learning Representations, (2024).

[2] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, *Generative adversarial nets*, in Advances in Neural Information Processing Systems, Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, eds., vol. 27, Curran Associates, Inc., (2014).

[3] J. Ho, A. Jain, and P. Abbeel, *Denoising diffusion probabilistic models*, in Advances in Neural Information Processing Systems, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, eds., vol. 33, Curran Associates, Inc., (2020), pp. 6840–6851.

[4] Z. N. Kesimoglu and S. Bozdag, *Supreme: A cancer subtype prediction methodology integrating multiomics data using graph convolutional neural network*, bioRxiv, (2022).

[5] D. P. Kingma and M. Welling, *An introduction to variational autoencoders*, Foundations and Trends® in Machine Learning, 12 (2019), p. 307–392.

[6] T. N. Kipf and M. Welling, *Semi-supervised classification with graph convolutional networks*, arXiv preprint arXiv:1609.02907, (2016).

[7] T. Ma and A. Zhang, *Affinity network fusion and semi-supervised learning for cancer patient clustering*, Methods, 145 (2018), pp. 16–24. Data mining methods for analyzing biological data in terms of phenotypes.

[8] K. Z. Nesibe and B. Serdar, *Graf: Graph attention-aware fusion networks*, arXiv preprint arXiv:2303.16781, (2023).

[9] C. Niu, Y. Song, J. Song, S. Zhao, A. Grover, and S. Ermon, *Permutation invariant graph generation via score-based generative modeling*, in Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics, S. Chiappa and R. Calandra, eds., vol. 108 of Proceedings of Machine Learning Research, PMLR, 26–28 Aug (2020), pp. 4474–4484.

[10] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, *The graph neural network model*, IEEE Transactions on Neural Networks, 20 (2009), pp. 61–80.

[11] H. Wang, J. Wang, J. Wang, M. Zhao, W. Zhang, F. Zhang, X. Xie, and M. Guo, *Graphgan: Graph representation learning with generative adversarial nets*, Proceedings of the AAAI Conference on Artificial Intelligence, 32 (2018).

[12] J. You, R. Ying, X. Ren, W. Hamilton, and J. Leskovec, *GraphRNN: Generating realistic graphs with deep auto-regressive models*, in Proceedings of the 35th International Conference on Machine Learning, J. Dy and A. Krause, eds., vol. 80 of Proceedings of Machine Learning Research, PMLR, 10–15 Jul (2018), pp. 5708–5717.