Instituto Tecnológico Autónomo de México

GENERACIÓN DINÁMICA DE GRAFOS CON GARANTÍAS DE PRIVACIDAD

Tesis

que para obtener el grado de

Maestro en Ciencias en Computación

Presenta

José Antonio Lechuga Rivera

Asesor

Dr. Carlos Fernando Esponda Darlington

Ciudad de México 2021

≪Con fundamento en los artículos 21 y 27 de la Ley Federal del Derecho de Autor y como titular de los derechos moral y patrimonial de la obra titulada "**GENERACIÓN DINÁMICA DE GRAFOS CON GARANTÍAS DE PRIVACIDAD**", otorgo de manera gratuita y permanente al Instituto Tecnológico Autónomo de México y a la Biblioteca Raúl Bailléres Jr., autorización para que fijen la obra en cualquier medio, incluido el electrónico, y la divulguen entre sus usuarios, profesores, estudiantes o terceras personas sin que pueda percibir por tal divulgación una contraprestación.≫

José Antonio Lechuga Rivera

Fecha

Firma

Dedicada a mis padres, "Ma" y "Pa", por permitir que durante tantos años mi única preocupación en la vida fuera cultivar mi conocimiento.

Agradecimientos

A mi mamá, por su amor incondicional y por mostrarme con su ejemplo a siempre dar lo mejor de mí para ayudar a los demás.

A mi papá, por ser mi modelo a seguir para ser un hombre bueno, amoroso y responsable, y por enseñarme a "barrerme en segunda" (lo que sea que eso signifique).

A mi esposa Karen, por enseñarme a disfrutar la vida como antes nunca pude, pero sobre todo, por no permitirme aburrir a nuestros conocidos platicando sobre los contenidos de esta tesis.

A mi hermana Cori, mi compañerita de toda la vida, por su amor infinito y por siempre creer que soy la persona más inteligente del mundo. No tiene pruebas, pero tampoco dudas.

A Salo, por su cariño, sus bromas y sus interesantes conversaciones. Espero podamos volvernos viejos platicando como hasta el día de hoy.

A mi padrino José Luis, por siempre inspirarme a ser un hombre tan culto y versátil como lo es él.

A los doctores Rubén Vásquez, Jorge Ibáñez y Gerardo Salinas, por contagiarme con su amor por la ciencia y la investigación.

Al Dr. Fernando Esponda, por confiarme su visión de la privacidad en grafos y por los cientos de consejos que permitieron que esta tesis fuera una realidad.

Resumen

Los avances en la teoría de grafos y el incremento en el poder de cómputo disponible, han permitido el estudio y simulación de sistemas complejos que antes estaban fuera del alcance tecnológico. A pesar de los diversos beneficios que esto ha traido a la sociedad, también ha vulnerado la privacidad de las poblaciones ya que las (ya no tan) complejas redes sociales en las que vivimos, son el caso de estudio perfecto para la teoría de grafos auxiliado por la gran capacidad de minar datos que provee la tecnología actual.

Aunque sí existe investigación previa sobre privacidad en grafos, el tema de garantizar la privacidad incluso ante la entidad generadora no ha sido ampliamente estudiado.

El presente trabajo presenta un mecanismo para efectuar la construcción de grafos, denominados *grafos ruidosos*, que permite mantener la privacidad de las conexiones entre vértices, a través de la introducción de aristas falsas, en todo momento durante la generación. El mecanismo propuesto realiza esta elaboración a través de la obtención individual de las listas de vecinos de cada vértice del sistema y va introduciendo ruido al grafo de manera que en ningún momento se tenga certeza completa de las aristas del mismo, incluso mientras está siendo generado.

Experimentalmente, este trabajo emplea grafos aleatorios generados de acuerdo al modelo de conexión preferencial de Barabási y Albert ya que, al día de hoy, es el modelo que mejor representa las redes que ocurren naturalmente.

Para medir la incertidumbre adicionada por el algoritmo propuesto, se emplea una métrica derivada de la entropía de Shannon que permite cuantificar el número de bits de información que el constructor del grafo debe obtener para conocer, con certeza absoluta, la lista de aristas reales de un vértice dado.

Finalmente, el algoritmo propuesto busca que los grafos ruidosos tengan la misma utilidad que un grafo que represente fielmente el sistema modelado, sobre todo en términos de la importancia relativa de los vértices con respecto a sus valores de las centralidades de grado, vectores propios, cercanía e intermediación. Para cuantificar esta utilidad, este trabajo calcula el coeficiente de correlación de Spearman sobre el ordenamiento de los vértices, con respecto a las métricas de centralidad mencionadas, entre el grafo ruidoso y un grafo tradicional sin incertidumbre.

Summary

Advances in graph theory and the increase in available computing power have allowed the study and simulation of complex systems that were previously out of technological reach. Despite the various benefits that this has brought to society, it has also endangered the privacy of populations since the (no longer so) complex social networks in which we live, are the perfect case study for graph theory aided by the great capacity for data mining provided by current technology.

Although there is previous research on graph privacy, the issue of guaranteeing privacy even from the graph-generating entity has not been adequately studied.

This paper presents a mechanism for the construction of graphs, called *noisy graphs*, that allows maintaining the privacy of the connections between vertices, through the introduction of fake edges, at all times during the generation. The proposed mechanism performs the construction through the individual collection of the neighbor lists of each vertex of the system, and introduces noise to the graph so that at no moment is there complete certainty of the edges of the graph, even while it is being generated.

Experimentally, this work employs random graphs generated according to the preferential attachment model of Barabási and Albert

since, to this day, it is the model that best represents naturally occurring networks.

To measure the uncertainty added by the proposed algorithm, a metric derived from Shannon's entropy is used to quantify the number of bits of information that the graph generator must obtain to know, with absolute certainty, the list of real edges of a given vertex.

Finally, the proposed algorithm aims for noisy graphs to have the same utility as a graph that exactly represents the modeled system, specially in terms of the relative importance of the vertices with respect to their values of degree, eigenvector, closeness, and betweenness centralities. To quantify this utility, this paper calculates the Spearman's rank correlation coefficient on the ordering of the vertices, with respect to the previously mentioned centrality metrics, between the noisy graph and a traditional graph without uncertainty.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

1.1.1 Complex Systems

Complex systems are those where a global state or behavior is impossible to obtain and/or predict from the isolated knowledge of its components. Humans have been studying these systems for thousands of years, simply because we are surrounded by structures that are hopelessly complicated [6]. To grasp the intuition behind this concept we can imagine a situation where we take a random classroom and swap the roles between the worst student and the teacher. Even if the entities composing the system are the same, the system's state will have drastically changed and will very likely have severe effects on the performance of the students as a consequence of the new interactions among its components.

As trivial as this example might seem, understanding complex systems, where many distinct elements interact in non-trivial patterns, is at the core of solving several of humanity's most important

problems and answering many of science's greatest questions.

Society itself, with the cooperation between billions of individuals, is the perfect illustration of a complex system, and to fully understand wealth redistribution, disease transmission or crime (among many other problems) is of paramount importance to model society as such.

Other examples [4, 6] are:

- **Health** – since 2001 we have an exhaustive list of all human genes, but to cure many diseases, such as cancer, we still need to accurately map how genes, proteins, metabolites and other cellular components interact with each other.

- **Neuroscience** – for over a century we have known that the neuron is the primary functional unit of the nervous system, nevertheless explaining reason and consciousness is still an impossible task as doctors would need to outline the trillions of neuron interactions in the mammalian brain.

- **Terrorism** – even if the term *terrorist* was coined during the French revolution, it is still a major international problem. Recently, a complex system approach has been used by anti-terrorism agencies to disrupt the financial network of terrorist organizations and map adversarial networks to uncover the role of their members and their capabilities.

Notwithstanding the diversity of complex systems, network science has demonstrated that behind each one, there is a graph that encodes the interactions between its components and that is driven by a set of common organizing principles. Ever since then, network scientists have been able to use graph theory to explore and predict these systems.

1.1.2 Privacy Concerns

Despite the numerous advantages that studying complex systems through graph theory can bring to society, the knowledge and capabilities offered by graphs can also be misused in ways that affect the privacy of organizations and individuals, because they reveal the mathematical laws that govern society at large [3]. One example of such exploitations is the social network mapping that the U.S. National Security Agency (NSA) performed through the tapping of communication systems.

But the problem is greater than only the U.S. having the ability to create social networks to exploit. Several other governments and big tech companies have also been collecting data of social interactions and relationships of millions of people [8, 9, 18]. As the evolution of mobile technology has enabled organizations to track human interaction up to the physical distance level, it is important to understand that we are losing the anonymity that our social interactions gave us before, because our complex social network is the perfect case study for graph theory.

Prior work on graph privacy (discussed in Chapter 3) has mainly focused on the problem of protecting privacy in published graph data [33]. This means researching ways to sanitize graph data before publishing it for analysis while retaining most of the original usefulness. These approaches are mainly concerned about adversaries that can intrude the privacy of people related to the graph once it is made public.

To the best of my knowledge, no previous research has worked on the ability of a collecting entity (i.e., a government or organization) to collect delicate relationships among a population. The present work affirms that some types of relationships, such as physical contact

between people, are too sensitive for any collecting organization, public or private, to possess with absolute certainty.

1.2 Objectives

1.2.1 Guaranteeing Graph Privacy

It is well known that guaranteeing privacy in graphs is extremely difficult because of the following reasons:

- There are many sources of information to protect such as vertices, edges, neighborhoods, induced subgraphs and their combinations [33].
- The information on the graph might be useful in a variety of different ways and because of this, different algorithms are needed to preserve different characteristics of the graph and most of them have antagonistic behavior on the final usefulness [33].
- Vertices and edges in a graph are correlated. A single change can have important consequences across the entire structure [21].
- It is difficult to quantify information loss on graphs [32].

Researchers working on the problem of preserving privacy for published graph data had to develop different privacy models, adversaries and graph-modification algorithms to overcome these challenges. Unfortunately none of their approaches have been able to solve all the problems at once [21]. It is extremely unlikely that a universal approach that solves all of these challenges for the problem of ensuring privacy from the graph-generating entity exists. Therefore, different solutions need to be proposed for each privacy model, adversary and graph use formulation.

1.2.2 Problem Definition

To properly define the problem that will be tackled in the present work, the following issues need to be formulated:

Usage of the Constructed Graph

This thesis focuses on maintaining vertex relative importance in terms of the following metrics: degree, betweenness, closeness and eigenvector centrality. Ordering vertices by these metrics can serve many purposes because they represent their relative importance for different network phenomena. For a brief explanation of these graph properties, please refer to Chapter 2.

As an example, a scheme for vaccine prioritization can be constructed by using the degree centrality metric as it would make sense to first vaccinate the most connected people in the social network.

Privacy Preservation

This work is concerned with ensuring privacy on sensitive network relationships. It is important to mention that these relationships are the ones responsible for the metrics that we want to maintain, so simply removing them is not an option for this problem formulation. To ensure edge privacy, while not being able to delete them, we aim to reduce the amount of certainty the collecting entity has on the edges of the graph it collected.

Further Restrictions

Finally, the proposed mechanism has two additional restrictions:

1. **False negatives** – we wish to ensure that no false negatives are given by our privacy-preservation approach, meaning that real edges are not removed from the original graph. The effect of this restriction is that even if the collecting entity cannot be sure whether a given edge exists or not in the real system the graph is representing (due to the uncertainty introduced to the graph), it is certain that no real connections are missing. This restriction is useful for many cases where false negatives introduce an important risk. An example of such a case is *COVID-19 Contact Tracing* where an authority cannot be certain of which individuals do have contact, but is sure it can inform every possible contagion as the set of real edges is a subset of the observed ones in the constructed graph.

2. **Uncertainty addition** – uncertainty must be introduced into the graph as each vertex is added. This means that at any given moment, even during the construction of the graph, there is no accurate information about graph edges. Without this restriction, many solutions could be proposed where the collecting entity first obtains an exact graph of the system it models and afterwards introduces noise to the graph to improve privacy. As it is reasonable to assume that the collecting entity will prefer to maximize the utility of the graph over the privacy of the data it contains (particularly since the present work proposes to protect the privacy of individuals even from the collecting entity itself), it is more sensible to place our trust in the construction algorithm rather than in the collecting entity.

1.3 Scope

The present work focuses on unweighted, undirected graphs constructed according to the Barabási–Albert preferential attachment model. This model is chosen as it is the one that best fits the degree distribution of natural occurring networks such as social networks [6].

In this thesis, a graph construction mechanism, which reduces the certainty that a collecting entity has about graph edges, is proposed. A graph constructed using the proposed mechanism will be denoted as a *noisy graph* throughout the present work. The proposed mechanism aims to retain most of the original graph's usefulness which is measured using the Spearman's rank correlation coefficient on the ordered vertices in terms of the centrality metrics previously mentioned.

As stated earlier, the proposed mechanism should not introduce false negatives into the graph edges and should ensure edge privacy even while the graph is under construction.

Chapter 2

Theoretical Framework

2.1 Introduction

Before delving into the related work or attempting to explain the construction of noisy graphs, it is important to familiarize the reader with the theory that will be used throughout this work. This chapter is in no way exhaustive in the topics it covers and should be considered as introductory content. If the reader wishes to look further into the topics addressed, he/she can refer to the citations provided throughout the chapter.

2.2 Graph Theory

2.2.1 Notation

A graph is a form of representing the relationships that occur between pairs of items. More formally, we can define a graph as a set of objects, called *vertices* or *nodes*, together with a collection of pairwise connections between them, called *edges* or *arcs* [15].

Mathematically a graph G is defined by the tuple $G = (V, E)$, where:

- V is the set $\{v_1, v_2, \ldots, v_n\}$ of all vertices in G
- $E = \{(v_i v_j) | v_i, v_j \in V\}$ is the set of edges in G
- $n = |V|$ is the number of vertices in G
- $e = |E|$ is the number of edges in G

Edges in a graph can either be *directed* or *undirected*. As the names suggest, the difference lies on whether the relationships contained in the graph have a sense of direction or not. For example, the relationships stored in Facebook can be represented with undirected edges as a friendship in that platform implies that both people involved have each other in their respective list of friends. In Instagram, directed edges are necessary because a user A following another user B does not imply that B also follows A.

Some other examples for undirected graphs are: physical contact, conversations and whether two people share a common ancestor. For directed graphs: payments, digital communications and road maps.

Additionally, edges can either be *weighted* or *unweighted*. Weighted edges are necessary when we wish to store a numeric value with the represented relationship instead of only acknowledging its existence. For example, a graph representation of phone conversations could store in every edge the number of phone calls two people have exchanged to obtain a weighted graph, or simply denote whether two people have ever had a phone call resulting in an unweighted one.

The remainder of this chapter will omit theory related to directed or weighted graphs as the presented mechanism only deals with *simple graphs* which are both unweighted and undirected.

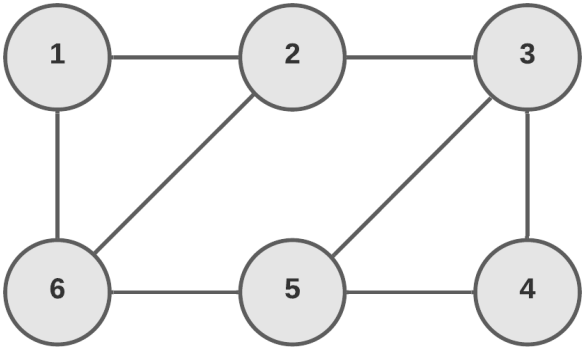


Figure 2.1. Simple Graph Visualization

Simple graphs are typically visualized by drawing the vertices as ovals and the edges as lines connecting pairs of ovals as depicted in Figure 2.1.

2.2.2 Data Structures for Graphs

There are two standard ways of representing a graph computationally: as a collection of adjacency lists, or as an adjacency matrix [12, 15].

The adjacency list representation consists of keeping a separate list for each vertex, containing every edge incident to it, meaning that there is an edge connecting this vertex with another one. This representation is very efficient when dealing with sparse graphs i.e., $e << n^2$. The adjacency-list representation for the graph in Figure 2.1 is shown in Table 2.1.

The adjacency-matrix representation, as its name suggests, maintains a particular matrix for a given graph. Each slot in the matrix is dedicated to storing a reference to the edge (i, j) for a

Table 2.1. Adjacency-List Representation

1	(1, 2), (1, 6)
2	(1, 2), (2, 3), (2, 6)
3	(2, 3), (3, 4), (3, 5)
4	(3, 4), (4, 5)
5	(3, 5), (4, 5), (5, 6)
6	(1, 6), (2, 6), (5, 6)

particular pair of edges i and j. This representation is more efficient for dense graphs i.e., where e approximates n^2.

Formally, the adjacency-matrix representation consists of an $n \times n$ matrix $A = (a_{ij})$ such that:

$$a_{ij} = \begin{cases} 1 & \text{if } (i,j) \in E \\ 0 & \text{otherwise.} \end{cases}$$

A summary of the algorithmic complexity of common graph operations, when using one of these two representation structures, is included in Table 2.2. This table was extracted from [15].

In Table 2.2 the term k_i is used to refer to the degree of vertex i. The degree of a vertex (for undirected graphs) simply means the number of incident edges it has. The degrees of the vertices of the graph in Figure 2.1 are shown in Table 2.3 as an example.

2.3 Random Graphs

Given the remarkable ability of graphs to represent real-life systems, scientists have been developing different methods to generate synthetic graphs that reproduce the properties of real-world networks [6]. These

Table 2.2. Algorithmic Complexity

Action	**Adjacency List**	**Adjacency Matrix**
Get number of vertices	$O(1)$	$O(1)$
Get number of edges	$O(1)$	$O(1)$
Get vertices	$O(n)$	$O(n)$
Get edges	$O(e)$	$O(e)$
Get edge (i, j)	$O(\min(k_i, k_j))$	$O(1)$
Get degree of i	$O(1)$	$O(n)$
Get edges of i	$O(k_i)$	$O(n)$
Insert vertex	$O(1)$	$O(n^2)$
Remove vertex i	$O(k_i)$	$O(n^2)$
Insert edge (i, j)	$O(1)$	$O(1)$
Remove edge (i, j)	$O(1)$	$O(1)$

Table 2.3. Vertex Degrees for Graph in Figure 2.1

k_1	k_2	k_3	k_4	k_5	k_6
2	3	3	2	3	3

methods have triggered important scientific breakthroughs as they allow us to study complex systems without the burden of collecting real data in the first place.

From a modeling approach, a network is a fairly simple object comprised of only vertices and edges. Nevertheless, the real problem is to decide where to place the edges between the vertices in order to reproduce the complexity of real systems [6].

The three most important models, for the purposes of the present work, will be described in this section.

2.3.1 Erdős-Rényi Model

In 1959, Erdős and Rényi developed the first model for generating random graphs with real properties of complex systems [13]. Their model consisted in defining N vertices and a probability p of connecting each pair of vertices [14].

2.3.2 Watts–Strogatz Model

Watts and Strogatz [29] provided an extension of the random graph model based on two observations:

1. **Small-World Property:** Most vertices can be reached from the others through a small number of edges [14]. In practice it has been shown that the average distance between two vertices depends logarithmically on n [6].

2. **High Clustering:** Many networks exhibit the occurrence of a significant amount of loops of size three [14]. Which implies that if vertex i is connected to vertices j and k, there is a high probability of vertices j and k being connected. As an example, for any pair of your friends, it is very likely that they are friends of each other as well.

Even though the Erdős-Rényi model exhibits the small-world property, it has a very small average clustering coefficient, which is difficult to achieve with a modification of their algorithm.

The intuition behind Watts-Strogatz graphs is to interpolate between a *regular lattice*, which has a high-clustering coefficient but does not show the small-world property, and a *random network* which has low clustering but exhibits the small-world property [6].

The construction of a Watts-Strogatz graph [14] starts with a ring graph (see Figure 2.2) of n vertices. Afterwards, each vertex in the ring is connected with its k nearest neighbors (or $k-1$ if k is odd). In Figure 2.2, v_1's four nearest neighbors would be the set $\{v_2, v_6, v_3, v_5\}$. Finally, each edge is randomly rewired with probability p. This rewiring process is performed by selecting an edge (i, j) and replacing it with probability p with a new one (i, w) where w is a graph vertex uniformly selected from $V \setminus \{i\}$.

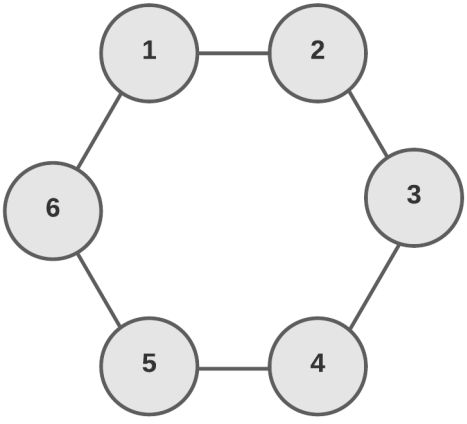


Figure 2.2. Ring Graph of Size 6

When $p = 0$ the presence of the small world phenomenon is absent, and when $p = 1$ it becomes an Erdős-Rényi graph so an intermediate value of p is necessary to achieve both properties.

2.3.3 Barabási–Albert Model

Barabási and Albert observed that while real-world networks exhibit some vertices with a disproportionately large amount of connections

[6], Erdős-Rényi and Watts-Strogatz models were unable to create graphs with highly connected vertices or hubs due to their construction mechanisms which enforce a Poisson degree distribution.

In reality, Barabási and Albert were the first people to observe hubs in a real-world network thanks to the development of the World Wide Web and the ability to study how one web site makes reference to another one. Nowadays, with the presence of social networks and *influencers*, this is an obvious phenomenon.

Based on this observation, the Barabási–Albert Model was proposed [5] in 1999 with a degree distribution following the power law.

The Barabási-Albert construction algorithm follows two basic rules: growth and preferential attachment [14]. The network is generated starting with a set of m_0 vertices ($m_0 < n$) that for random generated graphs usually takes the value of $m_0 = m + 1$. When a new vertex is added to the graph, it is connected with m other vertices ($m \leq m_0$) already present in the graph. The vertices that the new one is connected to are chosen following a linear preferential attachment rule, which means that the probability of the new vertex to connect with an existing vertex i, $\Pi(k_i)$, is proportional to the degree of i as Equation (2.1) shows.

$$\Pi(k_i) = \frac{k_i}{\sum_j k_j} \tag{2.1}$$

In Equation 2.1, k_i is the degree of the i-th vertex, and the sum represents the summation of the degrees of every preexisting vertex j.

2.3.4 Random Graphs Summary

Table 2.4 shows a quick comparison of the three random graph models discussed in this section.

Table 2.4. Random Graph Models

Measurement	Erdős-Rényi	Watts-Strogatz	Barabási-Albert
Degree distribution	Poisson	Poisson	Power Law
Small-world property	✓	✓	✓
High clustering	✗	✓	✓
Hubs	✗	✗	✓

2.4 Centrality Metrics

Since graphs represent a complexity that humans cannot process as is, various useful metrics have been developed to summarize different features of graph topology [23].

One question that naturally arises when working with complex systems is *how influential an element is within the system.* In other terms, this means we would like to know the importance of a vertex on a given graph structure. The measures that have been developed to answer this type of question are the *centrality metrics.* Now because "importance" can mean very different things in different contexts, it is not possible to have a universal metric of centrality, and we need to use different centrality metrics for different purposes.

In this section the degree, closeness, betweenness and eigenvector centrality metrics are presented as these are the ones that will be used by the present work.

2.4.1 Degree Centrality

The simplest centrality measure in a graph is just the degree of a vertex which is, as previously defined, the number of edges that are connected to it. Nevertheless, as a specific number of edges does not

tell us much without the actual graph size, this metric is usually normalized by dividing each degree by the number of total possible connections. This normalization is expressed mathematically by the following equation:

$$d_i = \frac{k_i}{n-1} \tag{2.2}$$

Where d_i is the normalized degree centrality metric for vertex i, k_i is the degree of this same vertex and n is the total number of vertices in the graph.

Although this might seem as a very simple metric, it is often very insightful. Newman [23] gives an excellent example concerning social networks which is that it seems reasonable to suppose that individuals who have connections with many others may have more influence, more access to information, or more prestige than those with fewer connections.

2.4.2 Eigenvector Centrality

The main problem with degree centrality is that it considers every connection equally important when one could argue otherwise as not all neighbors are equivalent. In most circumstances, the importance of a vertex in a graph is increased by having connections to other vertices that are important themselves [23]. This idea is what eigenvector centrality reflects. This metric gives each vertex a value proportional to the sum of the respective values of its neighbors.

To quickly grasp the intuition of this metric, we can use the graph in Figure 2.1 to work on an example. We start by proposing initial values $x_{i,0}$ for every vertex in the graph. For simplicity we can propose that $x_{i,0} = 1$ for all i. With these initial values it is possible to compute a

new set using Equation (2.3), where A is the adjacency matrix of the graph.

$$x_{i,n} = \sum_j A_{ij} x_{j,n-1} \tag{2.3}$$

After a first iteration i.e., $x_{i,1}$ what we are actually obtaining is the unnormalized degree centrality. If we normalize these values and afterwards repeat these two steps several times we start to see that the normalized values end up converging. These values have the property that they can be large because a given vertex has many neighbors, because it has important neighbors, or both.

The formula that we used to iteratively calculate the eigenvector centrality was first derived [7] in matrix notation as Equation (2.4) shows, where $\mathbf{A^n}$ means raising the adjacency matrix to the n-th power.

$$\mathbf{x_n} = \mathbf{A^n x_{n-1}} \tag{2.4}$$

It was also demonstrated [7] that the the convergence values for this metric, denoted simply as $\mathbf{x}$, satisfy Equation (2.5), where k_1 is the dominant eigenvalue (the eigenvalue with the highest absolute value) of the adjacency matrix.

$$\mathbf{A x} = k_1 \mathbf{x} \tag{2.5}$$

Returning to Newman's examples [23] on this topic again, in the case of social networks we could measure the importance of an individual using eigenvector centrality to balance the possibility that he/she might be important because he/she knows a lot of people, or just knows a few in very high places.

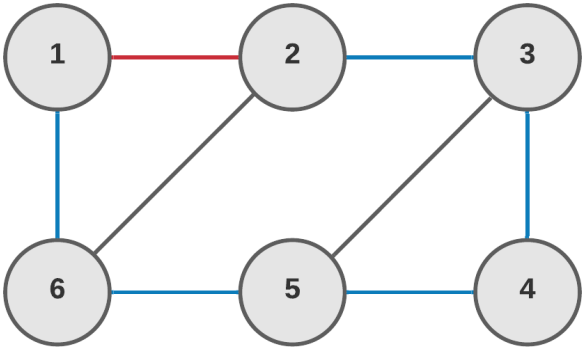


Figure 2.3. Geodesic Paths

2.4.3 Closeness Centrality

Another way of measuring the importance of a vertex in a graph is focusing on how close it is to all of the other vertices. To explain this metric it is important to first explain what a geodesic path is.

In graph theory, a path is simply a sequence of edges that join an array of vertices. They are normally used to determine which are the edges that would have to be *traversed* to get from one vertex to another. In the example shown in Figure 2.3, although there are others, two paths are highlighted, one in blue and one in red, by which one could travel from vertex 1 to vertex 2. A geodesic path, is simply the shortest path through a graph between two vertices [23]. Again, in Figure 2.3 of the two highlighted paths, the red one is the geodesic path between vertices 1 and 2. Any other path requires traversing at least two edges rather than just one.

Using d_{ij} to denote the length of the geodesic path from i to j (meaning the number of edges along the path), the closeness centrality,

l_i, is defined as the mean geodesic distance between vertex i and the rest of the n vertices in the network. Its mathematical expression is the following one:

$$l_i = \frac{1}{n} \sum_j d_{ij} \tag{2.6}$$

Where n is the total number of vertices in the graph.

This metric has small values for vertices that are separated from others by only a short geodesic distance on average. As Newman states [23], vertices with low closeness centrality might have better access to information from other vertices or more direct influence.

2.4.4 Betweenness Centrality

Concerning paths as well, betweenness centrality measures the degree by which a vertex lies on paths between other vertices. Specifically, this metric is defined as the number of geodesic paths a given vertex lies on [23].

Having a high value for this metric denotes the importance of a vertex. This is because on one hand, having a high betweenness gives it the ability to have more control over the information flowing through a graph. But also, vertex deletions with high values of betweenness are the most disruptive to graph communications. This phenomenon has very useful applications in utility networks such as internet, electricity and water.

Defining n_{st}^i as the following function:

$$n_{st}^i = \begin{cases} 1 & \text{if vertex } i \text{ lies on the geodesic path from } s \text{ to } t \\ 0 & \text{otherwise} \end{cases}$$

then, the betweenness centrality for vertex i is given by:

$$x_i = \sum_{st} n_{st}^i \tag{2.7}$$

Although this equation actually counts each geodesic path twice for undirected graphs (i.e., the geodesic path from s to t is the same path as the one from t to s), usually this does not matter as we only care about the relative magnitudes of this metric and the resulting vector is generally normalized.

As there might be more than one geodesic path between two vertices (in Figure 2.4 both the blue and red paths are geodesic paths for vertices 3 and 6), it is possible to express betweenness more generally by also defining g_{st} as the total number of geodesic paths from s to t adjusting the previous equation as follows:

$$x_i = \sum_{st} \frac{n_{st}^i}{g_{st}} \tag{2.8}$$

For this last equation, the term n_{st}^i/g_{st} is defined as zero by convention when $n_{st}^i = g_{st} = 0$.

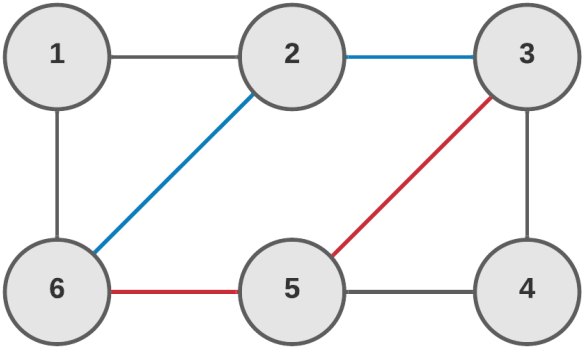


Figure 2.4. Multiple Geodesic Paths

Chapter 3

Related Work

3.1 Background on Graph Privacy

Most of the existing work on graph privacy deals with the problem of avoiding vertex re-identification in an anonymized public version of the graph. During the last two decades, information contained in social networks has become an important data source, and the scientific community has considered the release of such data as beneficial for the progress of science [11]. Nevertheless, they acknowledge that releasing this information can be harmful for the people, whose data is contained in the released graphs, as potential attackers may seek to use the published data to affect their privacy.

3.1.1 Naive Anonymization

Initially, to protect the privacy of individuals contained in the graph while preserving global properties (i.e., degree distribution, centrality metrics, spectral properties, aggregate queries, etc.), it was thought sufficient to perform a simple anonymization procedure. This

procedure consisted in stripping identifying vertex elements from the published graph such as name, email, address, social security number, etc. Backstrom *et al* [2] demonstrated that simple anonymization was not enough to prevent vertex re-identification if an attacker possessed some outside context of the graph. They experimentally demonstrated, on a 4.4-million-vertex social network, that the creation of 7 vertices by an attacker (before the anonymization and publishing of the graph data is done) compromised approximately 2,400 edge relations on average. Using the same network, they showed that the vast majority of individuals would be able to find themselves in a published graph, and thus discover the properties of the vertices they are linked to.

3.1.2 k-anonymity

Subsequent research [20, 32] focused on achieving *k-anonymity* based on various adversary knowledge such as neighborhoods [32] or vertex degree [20]. The task of graph k-anonymization consists on modifying a graph, by adding or removing edges, in order to construct a new one in which every vertex shares the same previously known property (i.e., graph structural property) with at least $k-1$ other vertices. Nonetheless, even if a specific vertex cannot be identified with a confidence greater than $\frac{1}{k}$, an attacker can still obtain sensitive information, as k-anonymization has two important shortcomings recognized in [22] which are:

1. **Homogeneity Attacks:** k-anonymity can create groups that leak information due to lack of diversity in the sensitive attribute. For example, if an attacker constrains a person's vertex in the graph (by means of a given graph property such as its degree, local neighborhood, centrality metrics, etc.) to a list of k vertices, but all of those vertices have HIV, then he/she has

already discovered a sensitive property of the vertex without pinpointing it.

2. **Background Knowledge Attack:** k-anonymity does not protect against attacks based on background knowledge. If in the previous example the possible diseases were Alzheimer, Turner Syndrome or HIV, and additionally it is known that the victim is a young man, then the attacker can be sure that the victim has HIV and perhaps even pinpoint him in the graph if no other vertex has this disease.

3.1.3 l-diversity

To solve these two problems, l-diversity was introduced [22] as a technique that improves upon k-anonymity. A given group of k vertices is said to be l-diverse if it contains at least l well represented values for every sensitive attribute S.

3.2 Taxonomy

Zhou *et al* [34], built a taxonomy for the privacy preserving methodologies on published graphs based on three concepts:

1. **Construction techniques**: this concept refers to the way the anonymized graph is constructed and is divided into the following two approaches:

   (a) *Clustering-based approaches*: [1, 10, 17, 31] they involve the clustering of vertices and/or edges into groups in order to anonymize a subgraph into a super vertex.

(b) *Noise addition approaches*: [27] they perform the anonymization task by inserting and/or deleting edges and/or vertices on the graph.

2. **Target attributes**: it refers to the attribute which the attacker wishes to reveal in the graph. This attribute can be:

   (a) Vertex existence

   (b) Vertex properties

   (c) Labels

   (d) Edges

   (e) Edge weights

   (f) Graph metrics

3. **Background knowledge**: this refers to the graph property that an attacker has of his/her victim. The candidate attributes are:

   (a) Identifying vertex attributes

   (b) Vertex degree

   (c) Edge type

   (d) Neighborhoods

   (e) Embedded subgraphs

   (f) Graph metrics

A matrix with all the possible combinations this taxonomy allows, is included in Table 3.1. In this table, the check mark denotes where the present work falls into this classification.

Table 3.1. Graph Privacy Taxonomy

Sensible property	Background Knowledge					
	Identifying attributes	Vertex degrees	Link relationships	Neighborhoods	Embedded subgraphs	Graph metrics
Vertex existence						
Vertex properties						
Sensitive vertex labels						
Link relationships	✓					
Link weight						
Sensitive edge labels						
Graph metrics						

3.3 Differentiation

As stated earlier, most of the existing research in graph privacy, like the ones described previously, are concerned about published graphs. Although there have been some other approaches to privacy such as [19], to the best of my knowledge, no one has tried to tackle the case where the privacy of the people in the graph is protected even from the graph-generating entity.

The main difference between the other research [16, 17, 20, 21, 30, 32, 34], and this one is that in the others, the privacy of the people involved in the graph is not protected in the original graph version. In such cases, there is absolute confidence that the entity in charge of generating the graph will not exploit it in unethical ways, and also that it has the ability to keep the original graph data safe from being stolen or bought. Nonetheless, in many cases the collecting entity cannot be trusted with these two guarantees and even less so when dealing with extremely sensitive properties such as physical contact between people.

The present work, in contrast to previous research, does not propose a mechanism to modify an original graph to a state that guarantees privacy, but rather seeks to ensure that there is never an unmodified graph and that the privacy of individuals is protected as soon as the

data is collected for its construction. In this way, trust is shifted from the organization in charge of generating the graph to the mechanisms used for its construction.

This new approach proposes new challenges, since it cannot leverage the use of global properties of the graph to maintain them, as they are never actually known to anyone. In other words, the construction algorithms have no knowledge of the actual value or state of the properties they seek to maintain during the privacy enforcement process. The present work proposes the first algorithm of this kind which is able to maintain the vertices' relative importance in terms of centrality metrics while ensuring privacy of sensitive connections between network vertices.

Chapter 4

Noisy Graph Construction Algorithm

In this chapter, the noisy graph model will be introduced along with the algorithm employed for its construction and its computational complexity. Also, the different metrics used to evaluate the performance of the designed algorithm, in terms of the objectives stated in Chapter 1, will be briefly explained. The results of the experiments are included in Chapter 5.

4.1 Model

A noisy graph is a simple graph where some of the edges included are not present in the original system that the graph is modeling. This paper refers to the edges originally present as *real edges* and the added ones as *fake edges*. Figure 4.1 shows an example of a noisy graph. In this figure, the black edges are meant to symbolize real edges (i.e., edges that are present in the real system) and red edges denote fake ones.

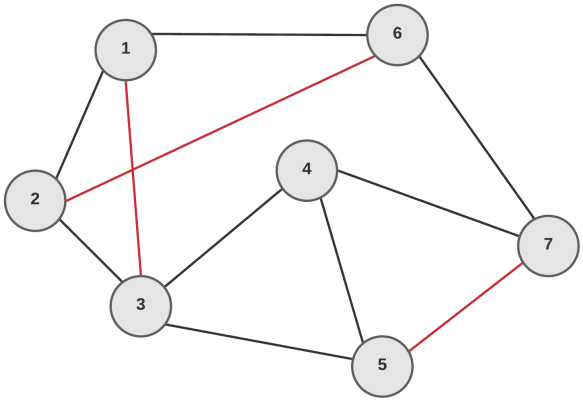


Figure 4.1. Noisy Graph Example

This difference is introduced here for demonstrative purposes, in a real implementation of a noisy graph there are no actual differences between the two types of edges.

In addition to the usual properties a simple graph has, a noisy graph, as defined in the present work, requires two extra properties:

- $\mathbf{G_{fr}}$ – is a global property which denotes the proportion of real to fake edges which is desired for each vertex. In the present paper, this proportion is used in the range [0, 1] where 0 means there are no fake edges and 1 means there are the same number of fake and real edges.

- $\mathbf{fr_i}$ – is the actual fake-to-real edge proportion that the i-th vertex has.

Mathematically fr_i is expressed as in Equation (4.1), where f_i means the number of fake edges of vertex i and r_i its corresponding number of real edges.

$$fr_i = \frac{f_i}{r_i} \tag{4.1}$$

A derived metric from these two is used in the construction algorithm, which is denoted as σ_i, by the present work. σ_i is a measurement of how compliant a given vertex is to the desired G_{fr} for the graph. Equation (4.2) expresses this metric mathematically.

$$\sigma_i = \frac{fr_i}{G_{fr}} \tag{4.2}$$

σ_i has the property of being less than one when fake edges are still missing (so that vertex i complies with the desired ratio), and being equal to one when it complies. It is worth mentioning that, in this case, regular optimization metrics such as the absolute difference or the squared difference are not useful as it is important for the algorithm to know if a vertex has enough fake edges or not, not just how far σ_i is from the target value.

As an example, Table 4.1 shows these metrics for the noisy graph in Figure 4.1 with a $G_{fr} = 0.5$.

vertex	$\mathbf{fr_i}$	$\sigma_\mathbf{i}$
1	0.50	1.00
2	0.50	1.00
3	0.33	0.66
4	0.00	0.00
5	0.50	1.00
6	0.50	1.00
7	0.50	1.00

Table 4.1. Noisy Graph Metrics for Figure 4.1 with G_{fr} of 0.5

Table 4.1 shows that in the graph depicted in Figure 4.1, vertices 3 and 4 are not compliant with the desired fake-to-real edge proportions G_{fr}.

4.2 Restrictions

Before explaining the algorithm, it is worth repeating the restrictions the stated objectives impose on its design:

1. The noisy graph should not present false negatives for the edges.
2. The real system global properties should never be known by anyone, not even the algorithm.
3. The certainty that one has about the edges in the graph should be reduced.
4. The usefulness of the graph, governed by the relative importance of its vertices in terms of the centrality metrics, should be preserved.

Meeting the first two constraints can be easily achieved and demonstrated from the design of the algorithm. Constraint 1 is attained by not deleting any real edge, this means that the only operation the algorithm can perform is the addition of fake edges.

Constraint 2 is ensured by not constructing an unmodified graph or keeping track of real edges for each vertex after noise has been added.

Constraint 3 is satisfied as long as fake edges are added to the graph. The difficult part of this restriction is really how to measure the information that has been removed from the graph by the construction algorithm. Regarding this concern, a simple metric derived from Shannon entropy [25] is introduced in this thesis to

measure the amount of information the collecting entity has lost for a given vertex. This metric is explained in Subsection 4.2.1.

Finally, it must be proved experimentally that the proposed algorithm complies with the fourth constraint. To do so, the ordering of vertices in terms of the previously discussed centrality metrics will be compared in both an unmodified system graph and the corresponding noisy graph by means of Spearman's rank correlation. The usage of Spearman's rank correlation is described in Subsection 4.2.2. Although the main objective of the present work is to maintain the relative importance of the vertices with respect to their centrality metric values, it is worth measuring how much the value distribution of those metrics is modified by the noisy graph construction algorithm. To perform this measurement Wasserstein distance will be used. It will enable us to understand how much the distributions are affected and if graphs of different sizes are affected equally. A brief explanation on how to calculate Wasserstein distance is included in Subsection 4.2.3. It is very important to emphasize that to make these comparisons an unmodified graph needs to be constructed. Nevertheless, this unmodified graph is only needed for these experimental purposes. An actual application absolutely does not need to maintain a graph without modifications.

4.2.1 Measuring Information Loss

Given that our proposal includes modifying graphs in order to preserve edge privacy, we wanted to be able to measure the amount of uncertainty the graph owner experiences for the edges of a specific vertex, once fake edges are added.

The intuition behind our metric is that the owner of the graph can know the amount of edges any given vertex has and, very likely, the

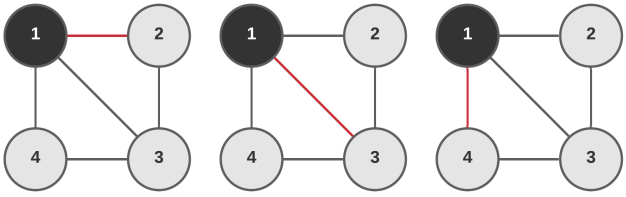


Figure 4.2. Combination Hypotheses Example

proportion of real to fake edges, but not which ones are real or fake. Because of this, the collector will need to gain some amount of knowledge in order to determine which of all the possible combination hypotheses corresponds for that given vertex. For example, in Figure 4.2 the graph owner might know that vertex 1 has one fake edge (depicted in red in this figure), but cannot determine which of the three presented graphs is the real one. The amount of information the owner needs to obtain to be able to discard all incorrect hypotheses can be calculated with Shannon's uncertainty measure [25] given by Equation (4.3).

$$S_i = -\sum_{j}^{H_i} p_j \log_2 p_j \tag{4.3}$$

In Equation (4.3) p_j is the probability that hypothesis X_j is true and H_i is the number of possible hypotheses for the i-th vertex. We assume all hypotheses are equally likely (i.e., $p_j = \frac{1}{H_i}$ for all j) so this equation can be simplified as shown in (4.4).

$$S_i = -\log_2 \frac{1}{H_i} \tag{4.4}$$

Calculating the number of hypotheses for the i-th vertex from the quantity of real and fake edges it has (r_i and f_i respectively), can be done by obtaining the binomial coefficient expressed by (4.5).

$$H_i = \binom{r_i + f_i}{f_i} \tag{4.5}$$

The final expression used to calculate the uncertainty introduced into a vertex is given by Equation (4.6).

$$S = -\log_2 \frac{1}{\binom{r_i+f_i}{f_i}} \tag{4.6}$$

It could be argued (given the mechanism presented in Section 4.3) that the graph owner will not be able to know the exact amount of real and fake edges for every vertex, but will only have an upper bound of fake ones ($\max_{f_i}$), obtained by means of the parameter G_{fr}. In this case, the total number of hypotheses, H_i, would be given by Equation (4.7), where k_i is the degree of vertex i.

$$H_i = \sum_{x=0}^{\max_{f_i}} \binom{k_i}{x} \tag{4.7}$$

Nevertheless, as in most cases the graph owner will actually know the exact number of fake and real edges, expression (4.5) will be the one used for our experimental reports as it provides a lower bound for the amount of hypotheses, and thus for the information loss.

4.2.2 Assessing Graph Utility

Before explaining the metric that will be used to determine the preservation of graph utility, it is important to clarify the problem that is being tackled. To make this explanation illustrative, an example is presented for the noisy graph in Figure 4.1. As stated earlier, we assess graph utility in terms of how much the original vertex ordering by centrality metrics is preserved in the noisy graph.

	Real		Noisy	
Vertex	Degree	Normalized	Degree	Normalized
1	2	0.33	3	0.5
2	2	0.33	3	0.5
3	3	0.50	4	0.67
4	3	0.50	3	0.5
5	2	0.33	3	0.5
6	2	0.33	3	0.5
7	2	0.33	3	0.5

Table 4.2. Vertex Ordering Example: Calculating Degree Centrality

Here the example of degree centrality is explained. Table 4.2, shows the degree centrality, as discussed in Chapter 2, for both the original graph (i.e., only using real edges) and the noisy graph (i.e., using real and fake edges) for the noisy graph in Figure 4.1.

With the calculated degree centrality, it is possible to order the vertices by their corresponding values having two potentially different orderings:

1. **Original ordering** – 3, 4, 1, 2, 5, 6, 7
2. **Noisy ordering** – 3, 1, 2, 4, 5, 6, 7

Note: it is worth mentioning that when two vertices have the same centrality metric value, their ordering is determined by their numeric identifier in ascending order. This mechanism helps us to remove the influence of when the vertex was inserted into the graph and only focus on their relative importance.

For this small set of vertices, it is possible to observe that some vertices have retained their relative importance, while other have not. For example, vertices 3, 5, 6 and 7 retained their relative importance (ordering) while vertices 1, 2 and 4 did not.

A way of transmitting this information concisely and that scales efficiently with graph sizes is Spearman's rank correlation coefficient which determines how well the relationship between the two orderings can be described using a monotonic function.

To calculate the Spearman's rank correlation coefficient between these two orderings, the following steps are necessary:

1. For every vertex in the graph, obtain its position of appearance in both orderings, i.e., rank.

2. For every vertex in the graph, calculate the squared difference of the ranks, D_i^2.

3. Compute Spearman's rank correlation coefficient according to Equation (4.8).

$$\rho = 1 - \frac{6 \sum D_i^2}{n(n^2 - 1)} \tag{4.8}$$

Equation (4.8) is an alteration of Pearson correlation coefficient where ranks are used in the original formula in place of the actual observations. This means that the correlation obtained does not concern the values themselves, but the ordering of such values. The mathematical derivation of (4.8) can be consulted in [26, 35]. For the purposes of the present work, in (4.8), n is the number of vertices in the graph. Table 4.3 performs these calculations for the ongoing example. Using the values for D_i^2 included in Table 4.3, $\sum D_i^2 = 6$ and so, $\rho = 0.893$.

Vertex	Ordering		Rank Differences	
	Real	Noisy	D_i	D_i^2
1	2	1	(2 - 1) = 1	1
2	3	2	(3 - 2) = 1	1
3	0	0	(0 - 0) = 0	0
4	1	3	(1 - 3) = -2	4
5	4	4	(4 - 4) = 0	0
6	5	5	(5 - 5) = 0	0
7	6	6	(6 - 6) = 0	0

Table 4.3. Vertex Ordering Example: Calculating Spearman's Rank Correlation Coefficient

For this metric, a value of 1 means that the graph has its utility intact, as defined for this work, while a value of 0 will mean that the graph has lost all of its utility.

4.2.3 Assessing Distribution Modification

Wasserstein distance is a very useful metric to assess the differences between two probability distributions. In this case, it can be used to determine how much the distributions of the original values for the centrality metrics were modified comparing the unaltered and the noisy versions of the graph.

Originally, Wasserstein distance for probability distributions P and Q was defined as Equation (4.9) shows [24].

$$W(P,Q) = \inf_{\pi \in \Gamma(P,Q)} \int_{\mathbb{R}^d \times \mathbb{R}^d} |X - Y| d\pi \tag{4.9}$$

Where $\Gamma(P,Q)$ is the set of all joint probability measures on $\mathbb{R}^d \times \mathbb{R}^d$ whose marginals are P and Q.

Nevertheless, it was demonstrated in [28] that this same distance could be calculated with the following expression:

$$W(P,Q) = \int_{-\infty}^{\infty} |F(x) - G(x)| dx \tag{4.10}$$

Where F and G are the distribution functions of the distributions P and Q respectively.

For the use case of the present work, as it is dealing with experimental observations, the integral in (4.10) can be computed with numerical integration methods using the empirical distribution functions constructed from the observed values.

Using the degree centrality values for both the original and the noisy graphs in Figure 4.1 (included in Table 4.2) we can obtain the empirical distribution functions for both the real and the noisy graphs. These functions are included in Table 4.4.

d	$F(d)$	$G(d)$	$\lvert F(d) - G(d) \rvert$
0.33	$\frac{5}{7}$	0	$\frac{5}{7}$
0.50	1	$\frac{6}{7}$	$\frac{1}{7}$
0.67	1	1	0

Table 4.4. Wasserstein Distance Computation Example

In Table 4.4, d represents the values of the degree centrality metric, F is the empirical distribution function for these values in the original graph and G the empirical distribution function for these values in the noisy version.

With these values we can compute the integral in (4.10) as a Riemann sum:

$$W_d = \frac{5}{7}(0.50 - 0.33) + \frac{1}{7}(0.67 - 0.50) = 0.146$$

4.3 Noisy Graph Construction Algorithm

4.3.1 Algorithm

The noisy graph data structure is defined as a collection of adjacency lists. More specifically, a hash table is used to store, for a given vertex (using its unique identifier as key), a set of all its neighbors. Additionally, three other maps are required (also using the vertex identifier as key) to store the amount of fake edges f, the amount of real edges r and the value of fake-to-real proportion compliance σ_i for every vertex in the graph.

Using these 4 maps, and a desired fake-to-real edge proportion for the graph (i.e., G_{fr}), a noisy graph can be constructed as described by Algorithm 1.

It is really important to emphasize that Algorithm 1 uses the concept of an "original graph" to simplify its nomenclature and be easier to understand. Nevertheless, the real graph is not necessary at all. What the algorithm requires from the real system is a neighbor list for every vertex. For example, in a social network, this means that the construction would involve interviewing one person at a time, and obtaining a list of all the other people with whom they have the desired relationship for the graph. As this algorithm introduces noise with every vertex insertion, an original graph is effectively unavailable after a second neighbor list is obtained.

4.3.2 Explanation

Line 1 of Algorithm 1 is in charge of initializing the necessary sets (hash tables) and arrays that will be needed during the construction. The nomenclature is as follows:

Algorithm 1 Noisy Graph Construction Algorithm

Input: Desired $\mathbf{G_{fr}}$ and original graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$

Output: Noisy graph $\mathbf{G_n} = (\mathbf{V_n}, \mathbf{E_n})$

1: $V_n \leftarrow \emptyset$, $E_n \leftarrow \emptyset$, $R \leftarrow \emptyset$, $F \leftarrow \emptyset$, $\Sigma \leftarrow \emptyset$
2: **for** $v_i \in V$ **do**
3: $V_n \leftarrow V_n \cup \{v_i\}$
4: $neighbors \leftarrow E(v_i)$
5: **for** $v_j \in neighbors$ **do**
6: $V_n \leftarrow V_n \cup \{v_j\}$
7: $E_n(v_i) \leftarrow E_n(v_i) \cup \{v_j\}$, $E_n(v_j) \leftarrow E_n(v_j) \cup \{v_i\}$
8: $R(v_i) \leftarrow R(v_i) + 1$, $R(v_j) \leftarrow R(v_j) + 1$
9: Update $\Sigma(v_i)$ and $\Sigma(v_j)$ according to Equation (4.2)
10: **end for**
11: **if** $\Sigma(v_i) < 1$ **then**
12: $n_f \leftarrow$ number of fake edges required to satisfy G_{fr}
13: $M \leftarrow V_n \setminus E_n(v_i)$ ▷ Set difference operation
14: Order M by increasing value of σ_i
15: **for** $v_j \in M$ **do**
16: **if** $F(v_i) \geq n_f \vee \Sigma(v_i) \geq 1 \vee \Sigma(v_j) \geq 1$ **then**
17: $i \leftarrow i + 1$
18: Go to For in line 2 ▷ Processes the following vertex
19: **end if**
20: $E_n(v_i) \leftarrow E_n(v_i) \cup \{v_j\}$, $E_n(v_j) \leftarrow E_n(v_j) \cup \{v_i\}$
21: $F(v_i) \leftarrow F(v_i) + 1$, $F(v_j) \leftarrow F(v_j) + 1$
22: Update $\Sigma(v_i)$ and $\Sigma(v_j)$ according to Equation (4.2)
23: **end for**
24: **end if**
25: **end for**

- $\mathbf{V_n}$ is the vertex set of the noisy graph.
- $\mathbf{E_n}$ is a hash table where for every $v_i \in V_n$ there is a set of its neighbors.
- $\mathbf{R}$ is a hash table storing the total number of real edges for every $v_i \in V_n$.
- $\mathbf{F}$ is a hash table storing the total number of fake edges for every $v_i \in V_n$.
- $\mathbf{\Sigma}$ is a hash table storing the corresponding σ_i for every $v_i \in V_n$.

Lines 2 through 10 mainly focus on adding the *interviewed* vertices and their corresponding real edges. Nevertheless, lines 8 and 9 are keeping score of the number of real edges each inserted vertex has, and correspondingly its fake-to-real edge proportion compliance (i.e., σ_i).

The rest of the algorithm deals with the addition of fake edges, and it is very important to notice that it happens after each vertex is *interviewed* which ensures that after the second vertex addition, the graph owner will lose edge certainty.

With the objective of ensuring that every vertex has a σ_i value of at least 1, while not having access to the original graph, a greedy approach is taken in lines 11 through 24. The first filter of this approach is that if a given vertex already has a σ_i greater than 1, it will not be processed by this part of the algorithm and so, no more fake edges will be added. This situation can happen as vertices are added to the graph even when they are not yet *interviewed*, but we get to know about their existence as a neighbor of an *interviewed* one.

Line 12 is in charge of calculating the amount of fake edges (n_f) the vertex that is being processed should have to comply with the specified

G_{fr}. This of course can be calculated deterministically by multiplying the number of real edges for the vertex by G_{fr}, but to prevent the algorithm from being completely deterministic, it is possible to obtain this number by means of a random process. This process could consist of obtaining r_i random numbers (i.e., the same amount as real edges for a given vertex) from a uniform distribution and setting n_f to the amount of those numbers whose values are greater than or equal to G_{fr}.

Afterwards, lines 13 and 14 are in charge of obtaining an array of possible fake neighbors, through a set difference operation, and ordering them increasingly in terms of their compliance metric. This ordering has two functions:

1. It ensures that vertices with lower fr_i are chosen first to have fake edges.

2. It allows the algorithm to exit early from the noise addition loop (lines 15 to 23) as if a fake neighbor candidate is seen with a σ_i value higher or equal to 1, we can be sure that the rest will satisfy this condition.

With this list of candidate fake neighbors, it is possible to start adding fake edges and keeping track of the noisy graph metrics, which is what lines 20 through 22 perform.

Finally, it is very important to notice that there is a set of three conditions (lines 16 to 19) that act as an early exit for an *interviewed* vertex. If any of the following conditions are met, the algorithm starts *interviewing* the next vertex in the system:

1. $F(v_i) \geq n_f$ – this means that the vertex that is being processed already has the desired amount of fake edges.

2. $\Sigma(v_i) \geq 1$ – this means that the processed vertex already complies with G_{fr}.

3. $\Sigma(v_j) \geq 1$ – this means that the fake neighbor candidate already complies with G_{fr}, and because of the imposed ordering, all of the remaining candidates do as well.

Although the first two conditions are similar, they are not always equivalent, specially when n_f is obtained through a random process.

4.3.3 Complexity

In this subsection, the time complexity of the algorithm will be analyzed. In order to make this process easier to understand, Table 4.5 details the time complexity for every part of Algorithm 1.

As mentioned in Table 4.5, the algorithm will at least have a complexity of $O(n)$ because of the outer *for loop* which represents the *interviewing* process of every vertex in the system. Of the inner operations, the dominant element is the ordering of the possible fake neighbors of the vertex, which has a complexity of $O(n \log n)$.

Taking this into consideration, then the noisy graph construction algorithm has a time complexity of $O(n^2 \log n)$.

4.3.4 Simple Example

To help the reader understand Algorithm 1 faster, the noisy graph construction from Figure 4.1 will be exemplified step by step in this subsection using a G_{fr} of 0.5. As before, real edges are drawn black, while fake ones are depicted in red.

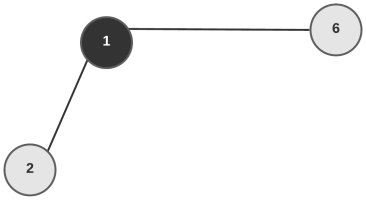
If we were to *interview* each vertex in Figure 4.1 about their real neighbors (i.e., connections) in increasing order of their numeric label,

Line	Complexity	Observations
1	$O(1)$	Hash tables and sets initialization
2	$O(n)$	Every vertex in the system must be interviewed
3	$O(1)$	Vertex insertion in adjacency list representation
4	$O(1)$	Getting neighbor set in hash table
5	$O(k_i)$	Every neighbor of vertex i is processed
6	$O(1)$	Vertex insertion in adjacency list representation
7	$O(1)$	Edge insertion in adjacency list representation
8	$O(1)$	Updating value in hash table
9	$O(1)$	Updating value in hash table
11	$O(1)$	Accessing value in hash table
12	$O(n)$	If done through random process, $O(1)$ otherwise
13	$O(n)$	Set difference in a hash table implementation
14	$O(n \log n)$	Ordering a list of comparable elements
15	$O(n)$	When the processed vertex only has one edge
16	$O(1)$	Accessing values in hash table
17	$O(1)$	Simple addition
18	$O(1)$	Break statement
20	$O(1)$	Edge insertion in adjacency list representation
21	$O(1)$	Updating value in hash table
22	$O(1)$	Updating value in hash table

Table 4.5. Algorithm 1 Complexity

the answers would be $1 : \{2, 6\}$, $2 : \{1, 3\}$, $3 : \{2, 4, 5\}$, $4 : \{3, 5, 7\}$, $5 : \{3, 4\}$, $6 : \{1, 7\}$ and $7 : \{4, 6\}$.

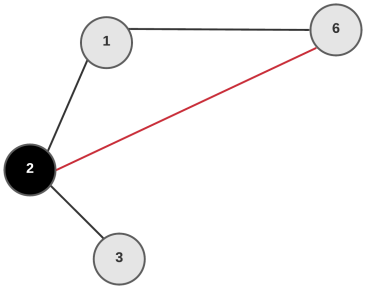
Here, the state of the noisy graph will be exemplified after interviewing each vertex:



	Before Noise		After Noise	
v_i	fr_i	σ_i	fr_i	σ_i
1	0	0	0	0
2	0	0	0	0
6	0	0	0	0

Figure 4.3. State After Interviewing Vertex 1

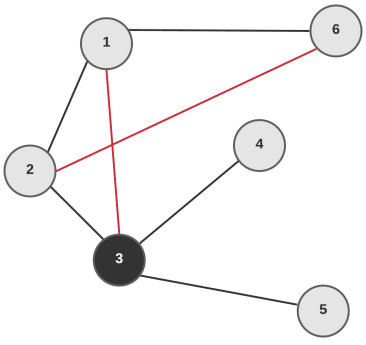
After obtaining the neighbors of vertex 1, in theory one fake edge should be added to satisfy the required G_{fr}. Nevertheless, as the list of possible fake neighbors is empty, no fake edges can be added with this first interview.



	Before Noise		After Noise	
v_i	fr_i	σ_i	fr_i	σ_i
1	0	0	0	0
2	0	0	0.5	1
3	0	0	0	0
6	0	0	1	2

Figure 4.4. State After Interviewing Vertex 2

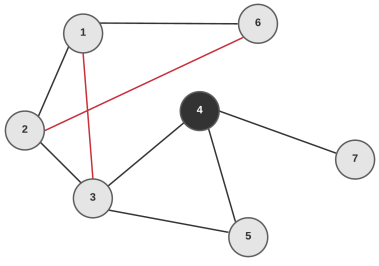
After obtaining the neighbors of vertex 2, we require 0.5 fake edges (i.e., 1 fake edge) to satisfy G_{fr}. The only possible fake neighbor is vertex 6, and as both vertex 2 and 6 have σ_i lower than one, the fake edge (2, 6) is added.



	Before Noise		After Noise	
v_i	fr_i	σ_i	fr_i	σ_i
1	0	0	0.5	1
2	0.5	1	0.5	1
3	0	0	0.33	0.67
4	0	0	0	0
5	0	0	0	0
6	1	2	1	2

Figure 4.5. State After Interviewing Vertex 3

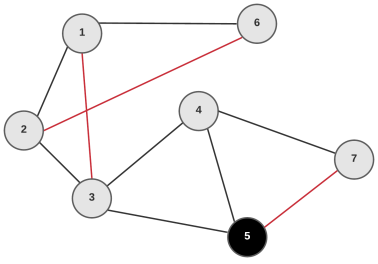
After obtaining the neighbors of vertex 3, we require 1.5 fake edges (i.e., 2 fake edges) to satisfy G_{fr}. The list of possible fake neighbors include vertices 1 and 6, nevertheless, as σ_6 is already ≥ 1, only the fake edge (1, 3) is added.



	Before Noise		After Noise	
v_i	fr_i	σ_i	fr_i	σ_i
1	0.5	1	0.5	1
2	0.5	1	0.5	1
3	0.33	0.67	0.33	0.67
4	0	0	0	0
5	0	0	0	0
6	1	2	1	2
7	0	0	0	0

Figure 4.6. State After Interviewing Vertex 4

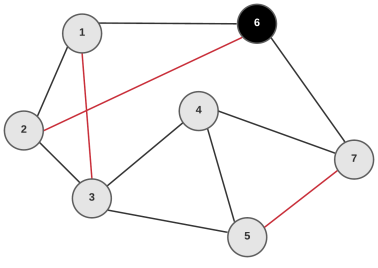
After interviewing vertex 4, we require 1.5 fake edges (i.e., 2 fake edges). Vertices 1, 2 and 6 are possible fake neighbors, nevertheless all of them already satisfy $\sigma_i \geq 1$ so no fake edges can be added.



	Before Noise		After Noise	
v_i	fr_i	σ_i	fr_i	σ_i
1	0.5	1	0.5	1
2	0.5	1	0.5	1
3	0.33	0.67	0.33	0.67
4	0	0	0	0
5	0	0	0.5	1
6	1	2	1	2
7	0	0	1	2

Figure 4.7. State After Interviewing Vertex 5

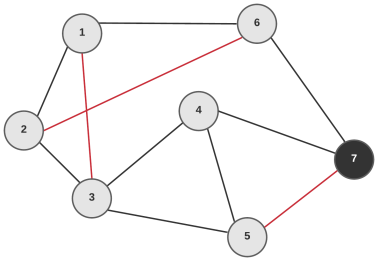
After interviewing vertex 5, 1 fake edge is required to comply. Vertices 1, 2, 6 and 7 are possible fake neighbors. As only σ_7 is ≤ 1, fake edge (1, 7) is the one that is added.



	Before Noise		After Noise	
v_i	fr_i	σ_i	fr_i	σ_i
1	0.5	1	0.5	1
2	0.5	1	0.5	1
3	0.33	0.67	0.33	0.67
4	0	0	0	0
5	0.5	1	0.5	1
6	0.5	1	0.5	1
7	0.5	1	0.5	1

Figure 4.8. State After Interviewing Vertex 6

After interviewing vertex 6 and adding real edge (6, 7), $\sigma_6 \geq 1$ is already true, so no fake edges are added.



	Before Noise		After Noise	
v_i	fr_i	σ_i	fr_i	σ_i
1	0.5	1	0.5	1
2	0.5	1	0.5	1
3	0.33	0.66	0.33	0.66
4	0	0	0	0
5	0.5	1	0.5	1
6	0.5	1	0.5	1
7	0.5	1	0.5	1

Figure 4.9. State After Interviewing Vertex 7

Interviewing vertex 7 adds no new real edges, and as $\sigma_7 \geq 1$ is already true, no fake edges are added. This is the final state of the graph. After the construction ended, only vertices 3 and 4 do not comply with the specified G_{fr} of 0.5. This situation is due to the limited number of vertices in the original system. Nevertheless, in real systems which involve a much greater amount of vertices, and usually deal with sparse networks, this situation is unlikely to appear as dramatically.

Chapter 5

Experimental Results

5.1 Experimental Conditions

As stated in Section 1.3, this work focuses on unweighted, undirected graphs constructed according to the Barabási-Albert model as it is the one that most closely represents natural occurring networks.

From the required inputs of the Barabási-Albert construction algorithm (see Subsection 2.3.3), and from the necessary parameters to construct our noisy graphs, there are in total three experimental variables that can be manipulated to study the behavior of the algorithm proposed by the present work. These variables are: n (i.e., graph size), m (i.e., new edges per inserted vertex), and G_{fr} (i.e., graph desired fake-to-real edges proportion).

In order to have a representative set of results to understand the behavior of noisy graphs with respect to the variations of these three parameters, the following ranges of variation were chosen:

- n – ten different graph sizes were chosen; specifically $n \in [100, 1000]$ in increments of 100 vertices.

- m – as the value of m must satisfy $1 \leq m \leq m_0 < n$, then ten different ratios of $\frac{m}{n}$ were chosen in the range $\frac{m}{n} \in [0.1, 1.0]$ in 0.1 increments.

- G_{fr} – this parameter was also chosen in the range $G_{fr} \in [0.1, 1.0]$ in 0.1 increments giving place to ten different configurations.

It should be pointed out that, given the restrictions of the Barabási-Albert construction algorithm, in the case where $\frac{m}{n} = 1$ as $m < n$ must be satisfied, the value of m, was set to $n - 1$.

These experimental conditions give rise to 1,000 noisy graphs that allow us to study the behavior of the targeted metrics. During the rest of this chapter, the results of G_{fr} compliance, graph uncertainty, and graph utility will be shown and discussed.

5.2 G_{fr} Compliance

As expressed by equation (4.2), σ_i compliance is a measure of how close the desired fake-to-real edge proportion is to the desired one by the graph constructor, i.e., G_{fr}. For the purpose of this research, it is desired that most of the vertices have a $\sigma_i \geq 1$. Nevertheless, the optimal value is 1 for any vertex as higher values of σ_i negatively affect graph utility while the desired privacy has already been achieved.

Figure 5.1 shows the results obtained for the average σ_i of the vertices for each of the experimental conditions. Before explaining the actual results, it is noteworthy that the cases where $\frac{m}{n} = 1.0$ (i.e., $m = n - 1$) behave very differently from the others. This is because normally, the initial m_0 vertices in a random Barabási-Albert graph (see Subsection 2.3.3) start from a star graph of $m + 1$ vertices. This

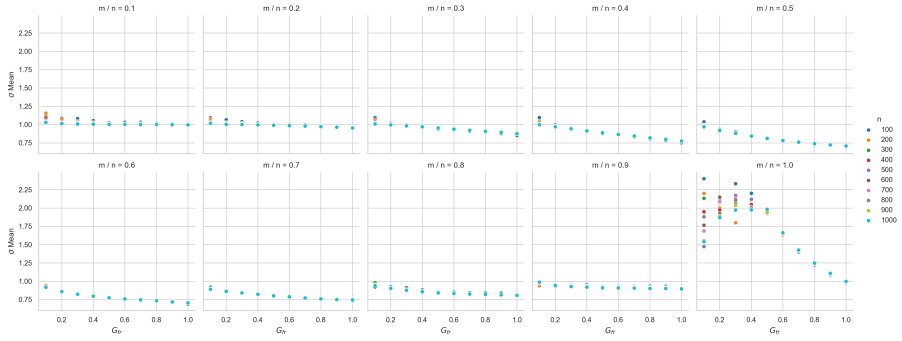


Figure 5.1. σ Mean (Complete Set)

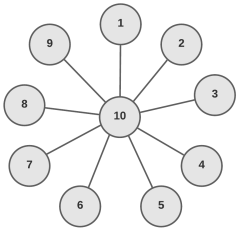


Figure 5.2. Star Graph with 10 Vertices

means that, in our case where $m = n - 1$, all n vertices will be part of the initial star graph.

As an example, for the case $\frac{m}{n} = 1$ and $n = 10$, a random generated Barabási-Albert graph will have the topology shown in Figure 5.2.

Because star graphs are not the interest of the present work, the rest of the results will exclude the $\frac{m}{n} = 1$ cases. This allows us to focus on real Barabási-Albert graphs. Figure 5.3 shows again the results for σ_i compliance, but without this last case.

A quite noteworthy observation is that the effect of graph size (n) on the mean (Figure 5.3) and variance (Figure 5.4) of σ is really weak and gets weaker as G_{fr} approaches 1. This result is indicative that the

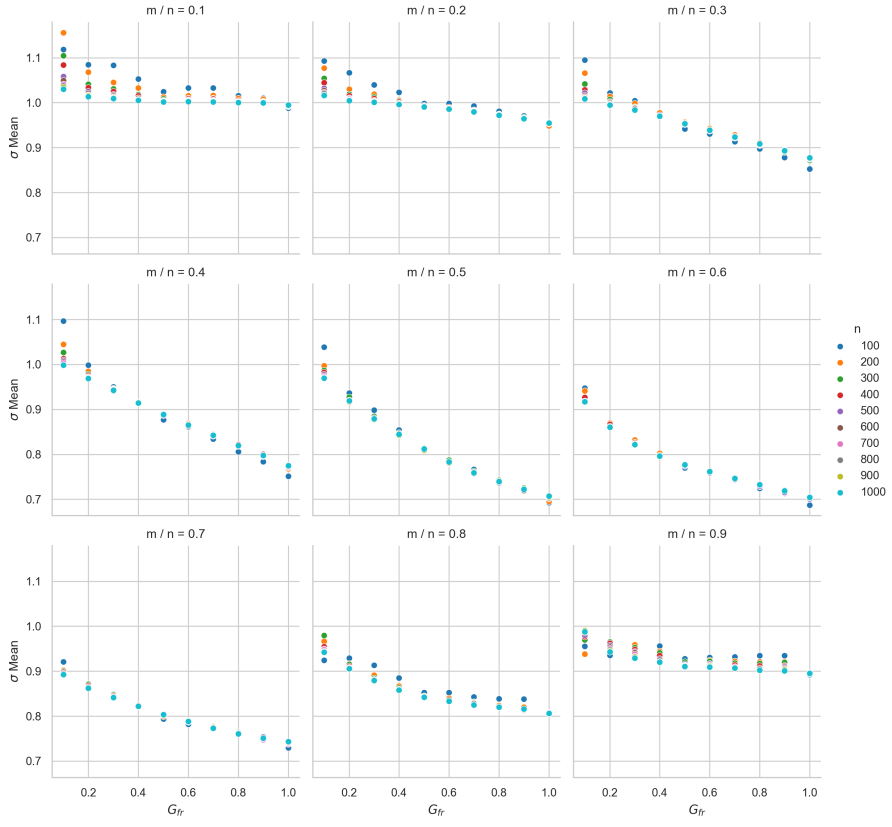


Figure 5.3. σ **Mean**

algorithm will perform similarly in terms of σ_i compliance even for much larger graphs.

Additionally, it can be observed that the mean σ value gets lower as G_{fr} increases, and the rate of change is greater in central values of $\frac{m}{n}$.

It is also noted that for some conditions a $\overline{\sigma} = 1$ is not obtained, especially for high G_{fr} requirements. This could be improved by a post-construction manipulation, where some vertices are allowed to have higher σ_i in order to ensure that $\sigma_i \geq 1, \forall i$. Nevertheless, this is out

of the scope of the present work and future work would need to assess the impact this post-construction alteration has on the graph utility as defined here. A quick solution so far would be to ask for a bigger fake-to-real edge proportion than required (i.e., $G'_{fr} > G_{fr}$) in cases where $\frac{m}{n}$ approaches 0.5 to ensure that $\overline{\sigma}$ tends towards 1.

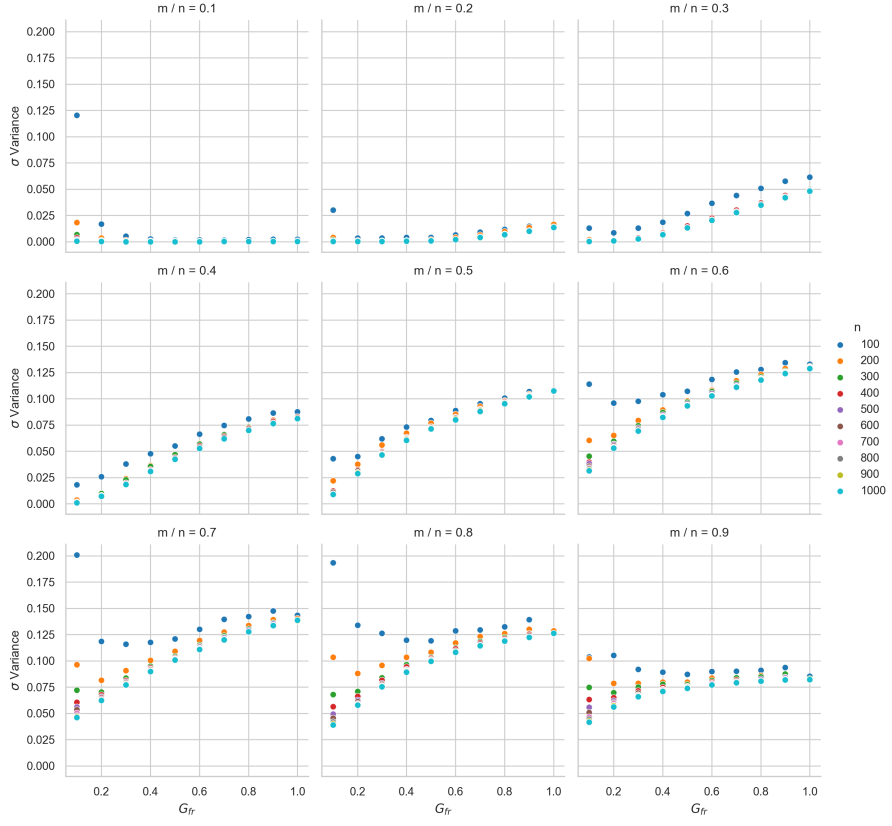


Figure 5.4. σ Variance

Finally, Figure 5.4 also shows that the variance of σ_i is greater for smaller graphs. This is perfectly expected as when the noisy graph construction algorithm has small sets of candidate fake neighbors, it is

difficult to achieve a $\sigma_i = 1$ for vertices with higher degrees.

5.3 Graph Uncertainty

As previously discussed in Subsection 4.2.1, Equation (4.6) defines uncertainty as the amount of bits (i.e., yes or no questions) the graph owner would have to obtain in order to completely know which edges for a given vertex are real and which ones are fake.

As expected, Figure 5.5 shows that the amount of uncertainty increases with the value of G_{fr}. Also, it must be noted that uncertainty also rises with graph size approaching values of approximately 700 bits of information for our biggest experimental cases. This means that on average, the graph owner would need to correctly answer 700 binary questions in order to perfectly determine which edges are fake for the average vertex.

Of course, once the fake edges of a given vertex is discovered, the amount of uncertainty for all its neighbors is reduced. It would be interesting for future research to determine the minimum amount of edge profiles that would need to be obtained to reduce the graph uncertainty to zero.

Mean uncertainty is greater for any n or G_{fr} when $\frac{m}{n}$ approaches 0.5. This observation is an effect that the power law has on the degree distribution for Barabási-Albert graphs. More specifically, extreme $\frac{m}{n}$ values sharpen the power law distribution which creates a situation where the majority of vertices have very few edges. Because of this, as previously noted, these extreme cases are easier to comply, but less uncertainty is added as a result of fewer fake edges being inserted to satisfy the desired ratio.

Figure 5.6 clearly shows that uncertainty variance is greater for

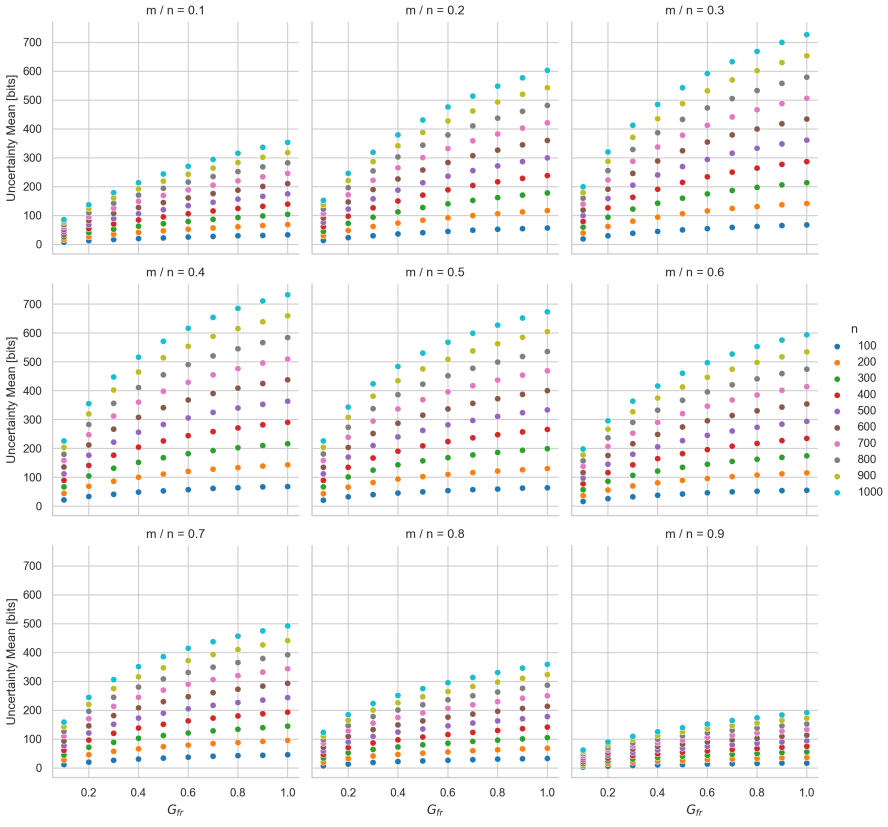


Figure 5.5. Uncertainty Mean

bigger graphs. Additionally, there is a decrease in the rate of change of variance versus G_{fr} that gets more critical as $\frac{m}{n}$ increases. This phenomenon gives place to a local maximum at lower values of G_{fr} as $\frac{m}{n}$ approximates 1. More experiments are required to determine if these observations are caused by opposing effect between these two parameters.

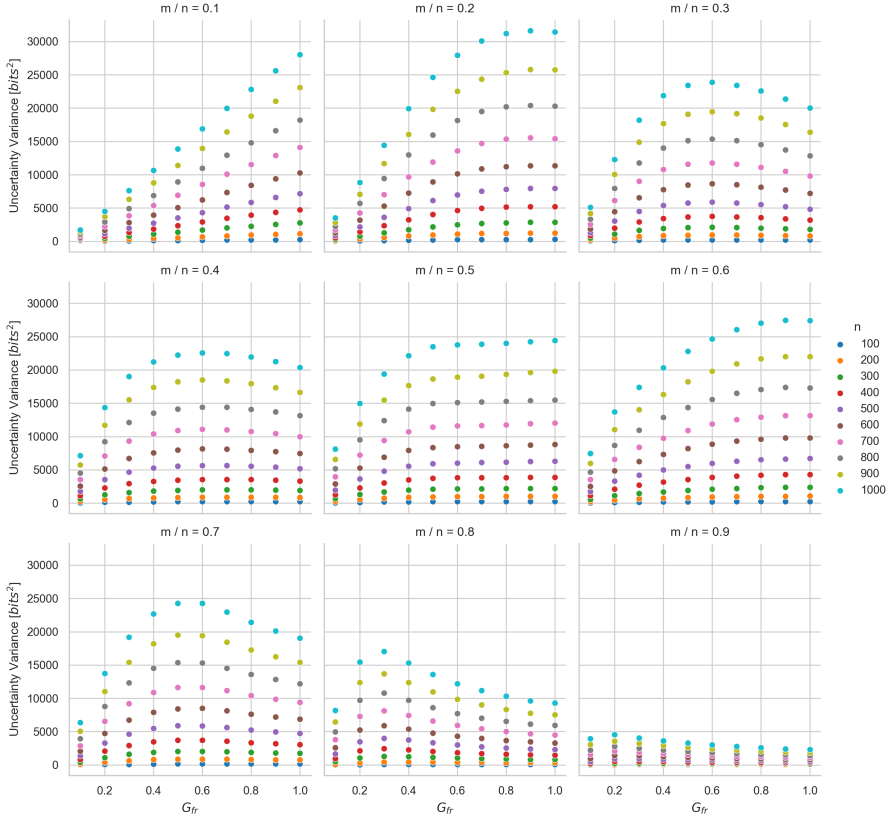


Figure 5.6. Uncertainty Variance

5.4 Graph Utility

After analyzing the privacy aspect of noisy graphs, it is now time to assess how well graph utility is preserved by the construction algorithm. For each of the centrality metrics used in this work (i.e., degree centrality, eigenvector centrality, closeness centrality, and betweenness centrality), the following two metrics are presented:

- **Spearman's** ρ – as explained in Subsection 4.2.2, this metric

will identify how much utility the graph has retained, as defined in this paper, as a result of being built by the noisy graph construction algorithm instead of just being an exact representation of the original system.

- **Wasserstein Distance** – as introduced in Subsection 4.2.3, this measurement helps us to determine how much the distributions of the original values for the centrality metrics were modified and whether these modifications are the same for different graph sizes.

5.4.1 Degree Centrality

Figure 5.7 shows that the degree centrality distribution is being modified in equal magnitudes regardless of the size of the graph. Also, as expected, it is perfectly clear that the distance between the degree centrality distribution of the unaltered graph and the noisy graph gets bigger as G_{fr} increases.

Figure 5.8 reveals that vertex ordering in terms of degree centrality is mostly unaltered for most of the experimental conditions. It is noteworthy that the Spearman's rank correlation coefficient is always over 0.88 and in many cases is 1.0, implying that the degree centrality sequence is not being affected by the noise introduced by the proposed construction algorithm.

This evidences that, even after significant amount of uncertainty has been added to the graph, the relative importance of the vertices with respect of their degree centrality is in many cases intact. This statement is especially true for graphs with $\frac{m}{n}$ values ≤ 0.5.

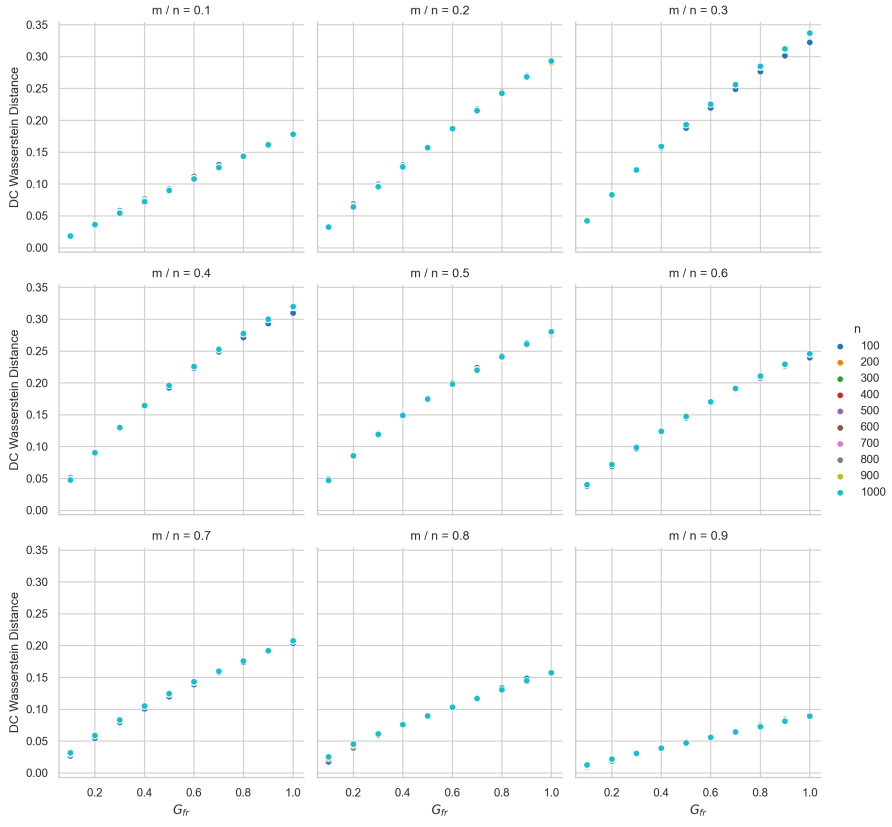


Figure 5.7. Degree Centrality Wasserstein Distance

5.4.2 Eigenvector Centrality

It can be observed in Figure 5.9 that, in the case of eigenvector centrality, there is a stronger modification of the distribution for smaller graphs. However, the same trend found in degree centrality is observed for the Wasserstein distance in the presence of variations of n and G_{fr}.

As in the case of degree centrality, the Spearman's rank correlation coefficient results were generally high for eigenvector centrality as well.

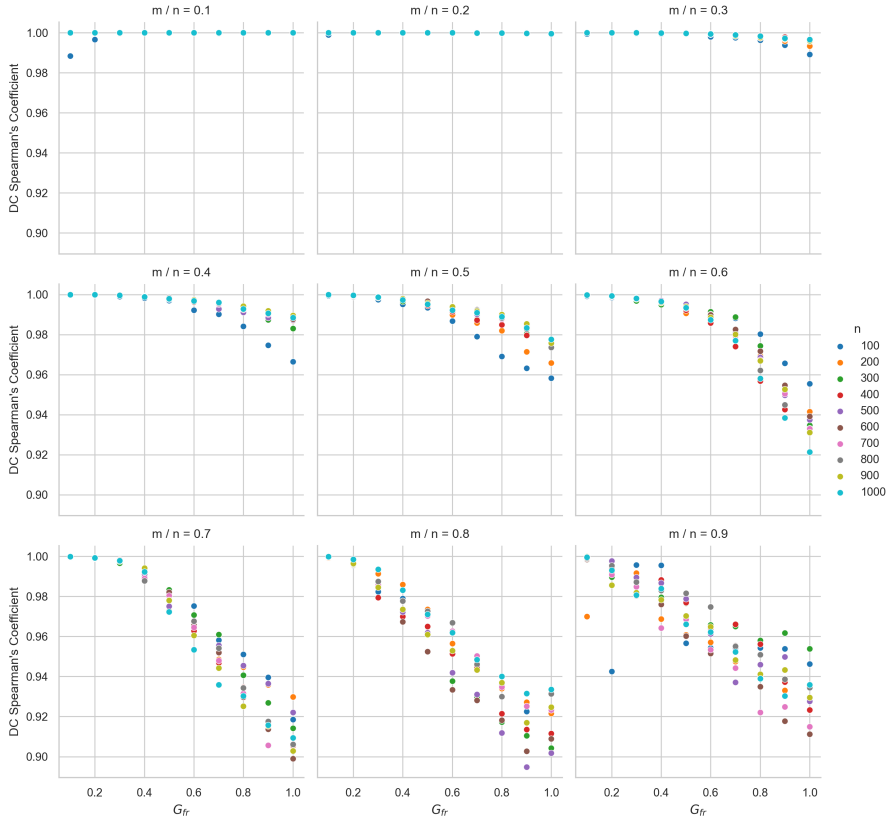


Figure 5.8. Degree Centrality Spearman's ρ

This is depicted in Figure 5.10 where it can be observed that this correlation coefficient is higher than 0.92 for all the cases with a value of $\frac{m}{n}$ lower than 0.5. Nevertheless, it must be mentioned that this metric starts degrading (falling under 0.90) for graphs with $\frac{m}{n} \geq 0.70$.

5.4.3 Closeness Centrality

The distribution of closeness centrality values is not affected to a greater extent for larger graph sizes as shown in Figure 5.11.

Figure 5.9. Eigenvector Centrality Wasserstein Distance

The results of the relative importance of the vertices with respect to closeness centrality values are excellent as it can be observed in Figure 5.12. Of all the experimental cases obtained, only two noisy graphs degrade the correlation coefficient slightly lower than 0.9, and these two cases are present for very high $\frac{m}{n}$ and G_{fr} values respectively.

As explained in the literature [23], geodesic distances usually increase logarithmically with graph size. This phenomenon might be why vertex ordering by closeness centrality is mostly unaffected as the

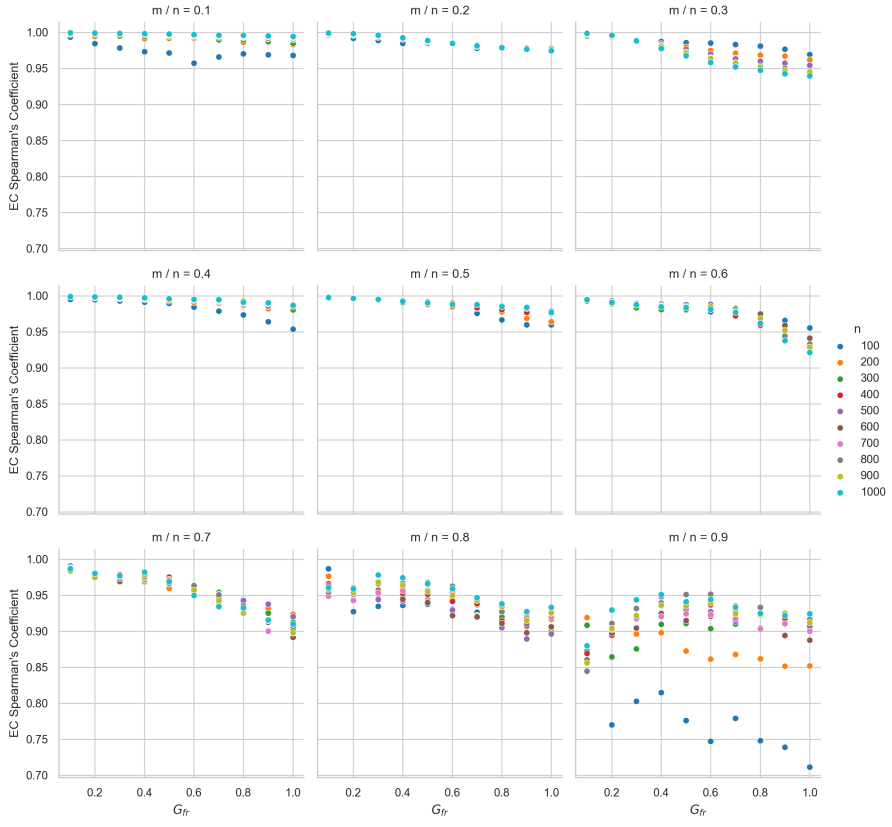


Figure 5.10. Eigenvector Centrality Spearman's ρ

proposed mechanism for noisy graph construction does not modify the graph size at all.

5.4.4 Betweenness Centrality

Finally, for the case of betweenness centrality, Figure 5.13 shows that once again its distribution is affected by the size of the graph being constructed. In spite of that, the general trend of Wasserstein distance values is maintained with respect to variations of the parameters $\frac{m}{n}$ and

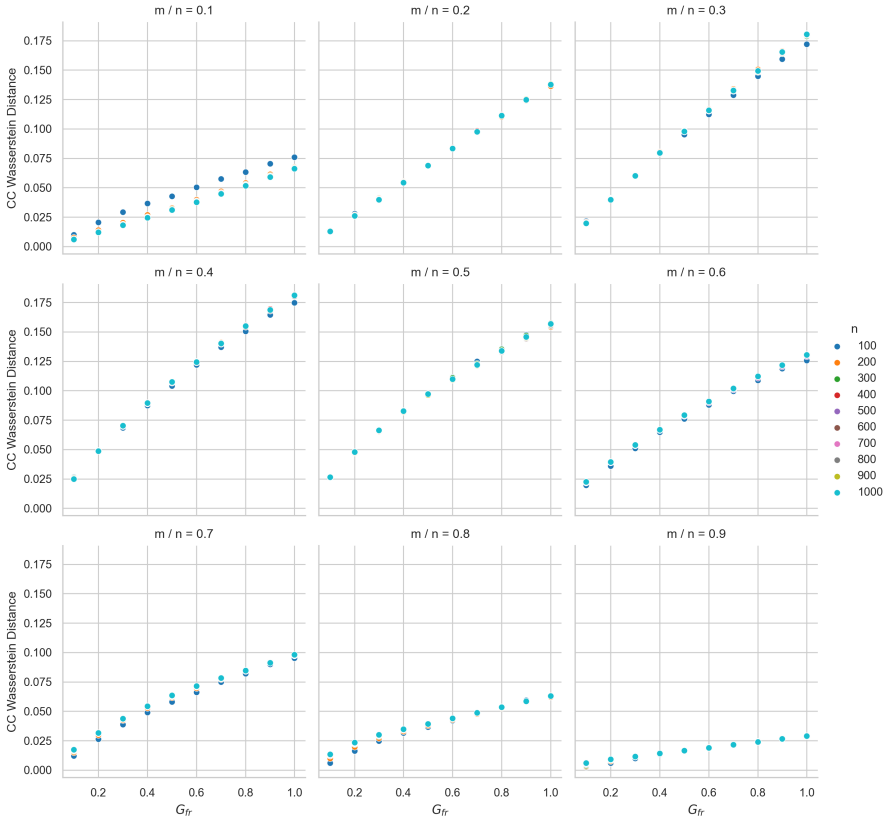


Figure 5.11. Closeness Centrality Wasserstein Distance

G_{fr}.

Although good correlation coefficients are found for experimental cases with $\frac{m}{n} < 0.7$, unfortunately experiments with values ≥ 0.7 show extreme degradation of Spearman's ρ. There are even some cases where a coefficient of zero is found, meaning that the proposed algorithm had the effect of randomly shuffling the graphs' vertices when ordered by betweenness centrality.

These last results are not surprising as the noisy graph construction

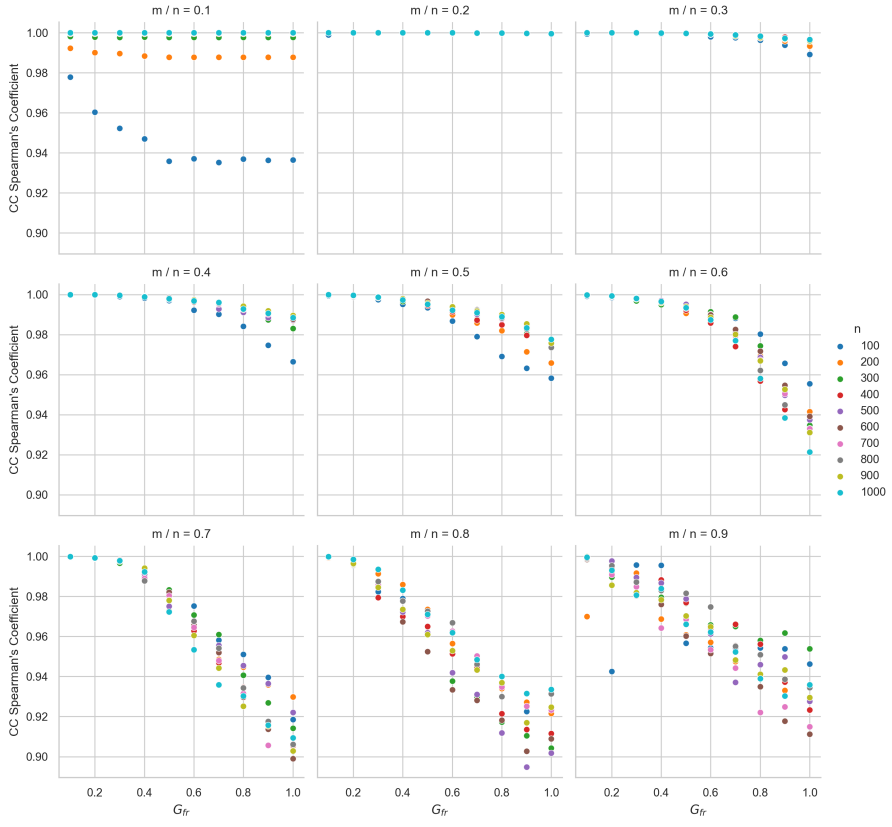


Figure 5.12. Closeness Centrality Spearman's ρ

mechanism adds a significant amount of edges. Although noisy graphs do not alter geodesic distance values, adding noise certainly modifies the number of geodesic paths between two vertices significantly, which is a relevant parameter for betweenness centrality computation.

5.4.5 Comparison

Figure 5.15 helps us examine the differences between centrality metrics in terms of Spearman's ρ. As observed, the noisy graph construction is

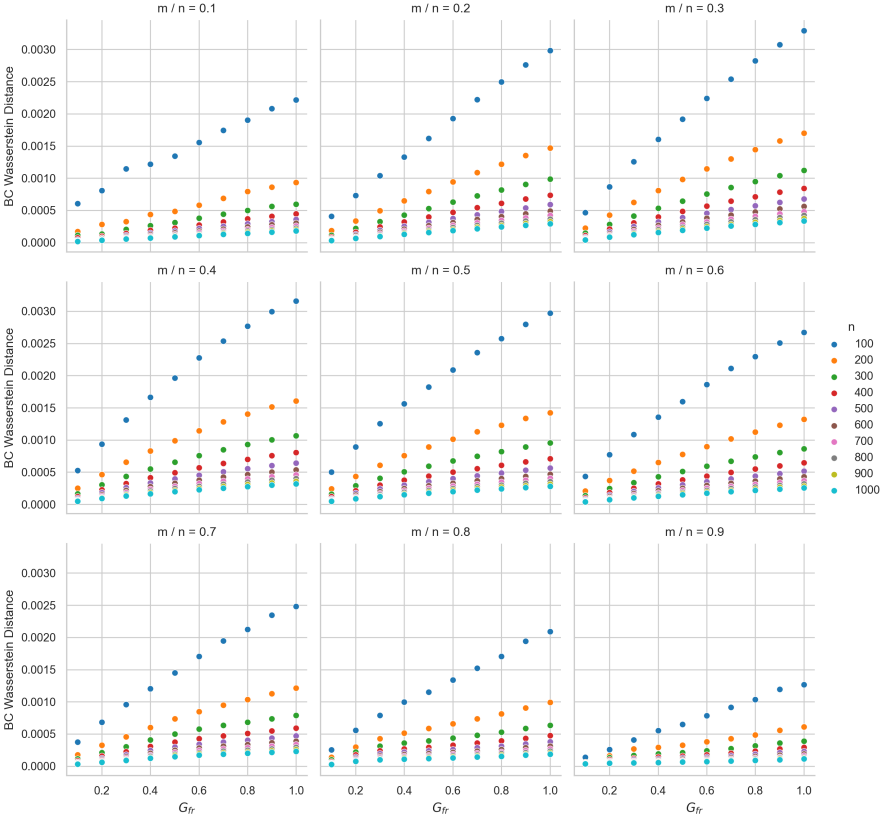


Figure 5.13. Betweenness Centrality Wasserstein Distance

able to maintain this measurement for degree, eigenvector, and closeness centralities with similar efficiency. For the case of betweenness centrality, noisy graphs definitely do not preserve vertex ordering for high $\frac{m}{n}$ values.

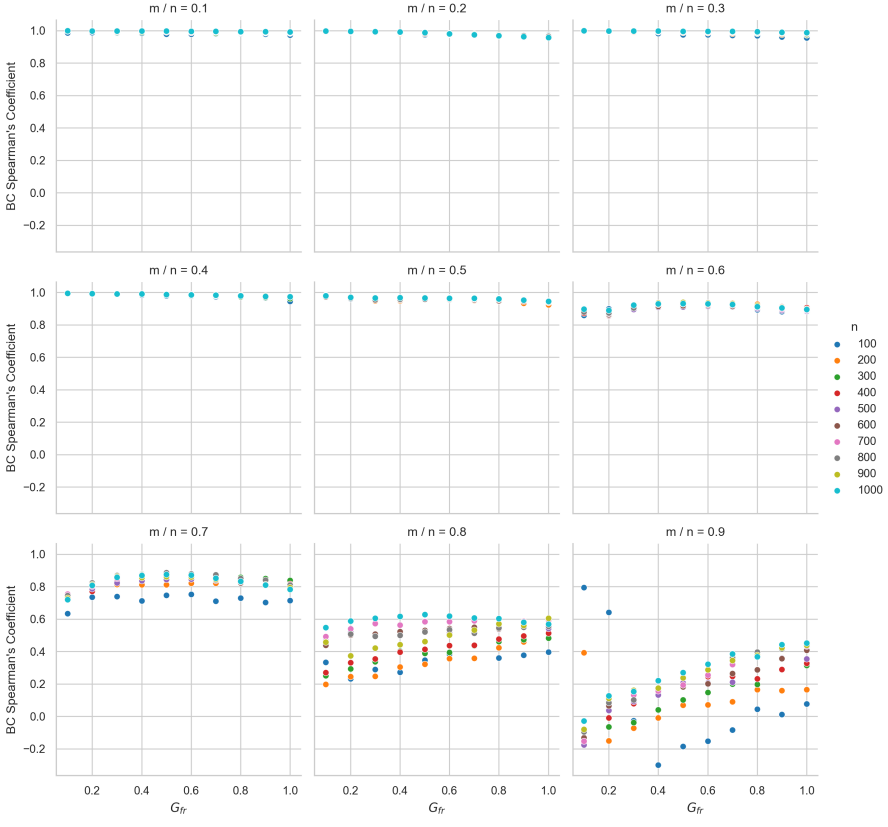


Figure 5.14. Betweenness Centrality Spearman's ρ

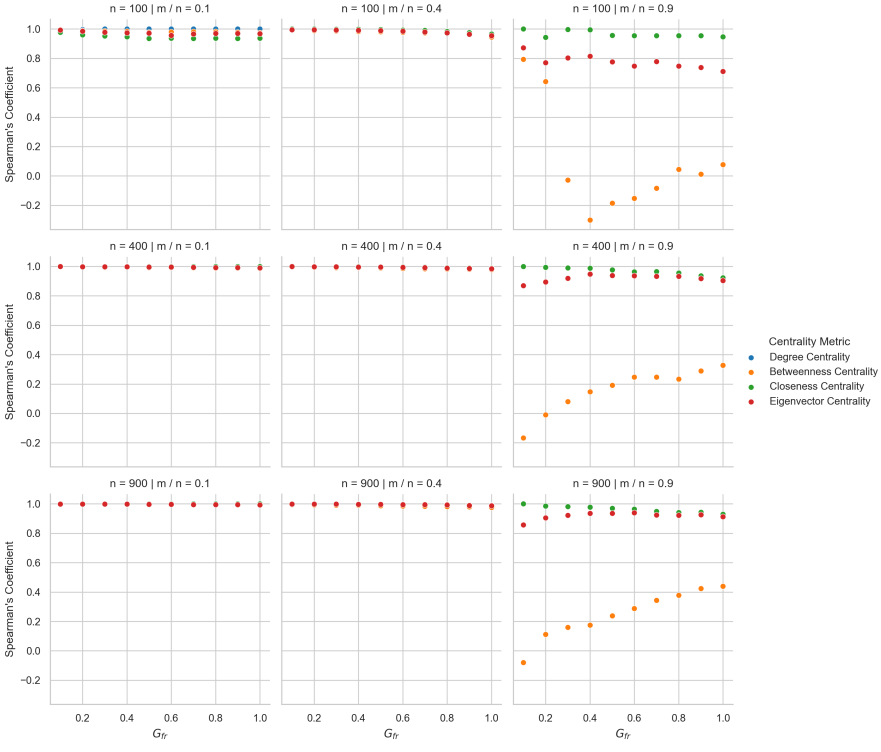


Figure 5.15. Spearman's ρ Comparison

Conclusions

This work presented a new type of graph, named noisy graph. This new type of graph is able to protect the privacy of sensitive relations between people even from the entity in charge of collecting the information used to construct the graph.

The results presented herein demonstrate that a new family of algorithms can exist where the privacy of the people involved in graphs is protected even from the entity in charge of generating them. In the present work, sensitive relations in a graph were protected while maintaining graph utility modeled as the vertex ordering by degree centrality, eigenvector centrality and closeness centrality. Nevertheless this is only one algorithm of the many that should exists to further protect people's privacy. As in the case of privacy in published graph data, different solutions need to be proposed for each privacy model and graph use formulation.

Future work in this privacy model and graph use focus, should be on three main topics: determine the amount of vertices' edge profiles that the graph owner should obtain to reduce the amount of uncertainty to zero, search for an algorithm that is able to preserve vertex ordering by betweenness centrality and develop a post-construction algorithm that improves G_{fr} compliance for edge cases.

References

[1] Korra S. Babu et al. “Anonymizing social networks: A generalization approach”. In: *Computers & Electrical Engineering* 39.7 (2013), pp. 1947–1961. ISSN: 0045-7906. DOI: https://doi.org/10.1016/j.compeleceng.2013.01.020. URL: http://www.sciencedirect.com/science/article/pii/S004579061300027X.

[2] Lars Backstrom, Cynthia Dwork, and Jon Kleinberg. “Wherefore art thou r3579x?: anonymized social networks, hidden patterns, and structural steganography”. In: *Proceedings of the 16th international conference on World Wide Web*. WWW ’07. Banff, Alberta, Canada: ACM, 2007, pp. 181–190. ISBN: 978-1-59593-654-7. DOI: 10.1145/1242572.1242598. URL: http://doi.acm.org/10.1145/1242572.1242598.

[3] Albert-László Barabási. *The ethical use of big data*. Oct. 2013. URL: https://www.politico.com/story/2013/09/scientists-must-spearhead-ethical-use-of-big-data-097578.

[4] Albert-László Barabási. “The network takeover”. In: *Nature Physics* 8.1 (Dec. 2011), pp. 14–16. ISSN: 1745-2481. DOI:

10 . 1038 / nphys2188. URL: http://dx.doi.org/10.1038/nphys2188.

[5] Albert-Laszlo Barabási and Albert Reka. "Emergence of Scaling in Random Networks". In: *Science* 286 (Nov. 1999), pp. 509–512. DOI: 10.1126/science.286.5439.509.

[6] Albert-László Barabási and Márton Pósfai. *Network science*. Cambridge: Cambridge University Press, 2016. ISBN: 978-1-107-07626-6. URL: http://barabasi.com/networksciencebook/.

[7] Phillip Bonacich. "Power and Centrality: A Family of Measures". In: *American Journal of Sociology* 92.5 (1987), pp. 1170–1182.

[8] Francesco Bonchi et al. "Social Network Analysis and Mining for Business Applications". In: *ACM Trans. Intell. Syst. Technol.* 2.3 (May 2011). ISSN: 2157-6904. DOI: 10.1145/1961189.1961194. URL: https://doi.org/10.1145/1961189.1961194.

[9] William J. Bradshaw et al. "Bidirectional contact tracing could dramatically improve COVID-19 control". In: *Nature Communications* 12.232 (2021), p. 9. DOI: 10.1038/s41467-020-20325-7.

[10] Alina Campan and Traian M. Truta. "Data and Structural k-Anonymity in Social Networks." In: *PinKDD*. Ed. by Francesco Bonchi et al. Vol. 5456. Lecture Notes in Computer Science. Springer, 2008, pp. 33–54. ISBN: 978-3-642-01717-9. DOI: 10.1007/978-3-642-01718-6_4. URL: http://dblp.uni-trier.de/db/conf/kdd/pinkdd2008.html#CampanT08.

[11] James Cheng, Ada Fu, and Jia Liu. “K-isomorphism: Privacy preserving network publication against structural attacks”. In: Jan. 2010, pp. 459–470. DOI: 10.1145/1807167.1807218.

[12] Thomas H. Cormen et al. *Introduction to Algorithms*. en. Third. MIT Press, July 2009. ISBN: 978-0-262-03384-8. URL: https://mitpress.mit.edu/books/introduction-algorithms-third-edition.

[13] Paul Erdős and Alfréd Rényi. “On Random Graphs I”. In: *Publicationes Mathematicae (Debrecen)* 6 (1959), pp. 290–297.

[14] Luciano da F. Costa et al. “Characterization of Complex Networks: A Survey of measurements”. In: *Advances in Physics* 56 (Jan. 2007), pp. 167–242. DOI: 10.1080/00018730601170527.

[15] Michael T. Goodrich, Roberto Tamassia, and Michael H. Goldwasser. *Data Structures and Algorithms in Java*. Cambridge: Wiley, 2014. ISBN: 978-1-118-77133-4.

[16] Michael Hay et al. “Anonymizing Social Networks”. In: *Technical Report 07-19* (2007).

[17] Michael Hay et al. “Resisting structural identification in anonymized social networks”. In: *PVLDB* 1 (Aug. 2008), pp. 102–114. DOI: 10.14778/1453856.1453873.

[18] Rizwana Irfan et al. “A survey on text mining in social networks”. In: *The Knowledge Engineering Review* 30.2 (2015), pp. 157–170. DOI: 10.1017/S0269888914000277.

[19] Aleksandra Korolova et al. “Link Privacy in Social Networks”. In: Apr. 2008, pp. 289–298. DOI: 10.1109/ICDE.2008.4497554.

[20] Kun Liu and Evimaria Terzi. “Towards Identity Anonymization on Graphs”. In: Jan. 2008, pp. 93–106. DOI: 10.1145/1376616.1376629.

[21] Kun Liu et al. “Privacy-Preserving Data Analysis on Graphs and Social Networks.” In: *Next Generation of Data Mining*. Ed. by Hillol Kargupta et al. Chapman and Hall / CRC Data Mining and Knowledge Discovery Series. CRC Press / Chapman and Hall / Taylor & Francis, 2008. ISBN: 978-1-4200-8586-0. URL: http://dblp.uni-trier.de/db/books/collections/KHYMK2008.html#0001DGK08.

[22] Ashwin Machanavajjhala et al. “L-diversity: Privacy beyond k-anonymity.” In: *ACM Trans. Knowl. Discov. Data* 1.1 (2007), p. 3. URL: http://dblp.uni-trier.de/db/journals/tkdd/tkdd1.html#MachanavajjhalaKGV07.

[23] Mark Newman. *Networks: an introduction*. Oxford; New York: Oxford University Press, 2010. ISBN: 9780199206650. URL: http://www.amazon.com/Networks-An-Introduction-Mark-Newman/dp/0199206651/ref=sr_1_5?ie=UTF8&qid=1352896678&sr=8-5&keywords=complex+networks.

[24] Aaditya Ramdas, Nicolas Garcia, and Marco Cuturi. “On Wasserstein Two Sample Testing and Related Families of Nonparametric Tests”. In: *Entropy* 19 (Sept. 2015), pp. 1–18. DOI: 10.3390/e19020047.

[25] Claude E. Shannon. “A Mathematical Theory of Communication”. In: *The Bell System Technical Journal* 27 (1948), pp. 379–423. URL: http://plan9.bell-labs.com/cm/ms/what/shannonday/shannon1948.pdf.

[26] Charles Spearman. “The Proof and Measurement of Association Between Two Things”. In: *American Journal of Psychology* 15 (1904), pp. 88–103.

[27] Vicenç Torra and Julián Salas. “Graph Perturbation as Noise Graph Addition: A New Perspective for Graph Anonymization”. In: Sept. 2019, pp. 121–137. ISBN: 978-3-030-31499-6. DOI: 10.1007/978-3-030-31500-9_8.

[28] S. S. Vallander. “Calculation of the Wasserstein distance between probability distributions on the line”. In: *Theory Probab. Appl.* 18 (4 1973), pp. 784–786. DOI: 10.1137/1118101.

[29] Duncan J. Watts and Steven H. Strogatz. “Collective dynamics of ‘small-world’ networks”. In: vol. 393. 1998, pp. 440–442. DOI: https://doi.org/10.1038/30918.

[30] Xintao Wu et al. “A Survey of Privacy-Preservation of Graphs and Social Networks”. In: Boston, MA: Springer US, 2010, pp. 421–453. ISBN: 978-1-4419-6045-0. DOI: 10.1007/978-1-4419-6045-0_14.

[31] Elena Zheleva and Lise Getoor. “Preserving the Privacy of Sensitive Relationships in Graph Data”. In: vol. 4980. Aug. 2007, pp. 153–171. ISBN: 978-3-540-78477-7. DOI: 10.1007/978-3-540-78478-4_9.

[32] Bin Zhou and Jian Pei. “Preserving Privacy in Social Networks Against Neighborhood Attacks.” In: *ICDE*. Ed. by Gustavo Alonso, José A. Blakeley, and Arbee L. P. Chen. IEEE Computer Society, 2008, pp. 506–515. ISBN: 978-1-4244-1836-7. URL: http://dblp.uni-trier.de/db/conf/icde/icde2008.html#ZhouP08.

[33] Bin Zhou, Jian Pei, and Wo-Shun Luk. "A brief survey on anonymization techniques for privacy preserving publishing of social network data." In: *SIGKDD Explor.* 10.2 (2008), pp. 12–22. URL: http://dblp.uni-trier.de/db/journals/sigkdd/sigkdd10.html#ZhouPL08.

[34] Bin Zhou, Jian Pei, and WoShun Luk. "A Brief Survey on Anonymization Techniques for Privacy Preserving Publishing of Social Network Data". In: *SIGKDD Explorations* 10 (Dec. 2008), pp. 12–22. DOI: 10.1145/1540276.1540279.

[35] Daniel Zwillinger and Stephen Kokoska. *CRC Standard Probability and Statistics Tables and Formulae*. Mar. 2000. ISBN: 1-58488-059-7.