# Master Computer Science

Measuring Structural Anonymity in Complex Networks

Universiteit Leiden


| | |
|---|---|
| Name: | Rachel G. de Jong |
| Student ID: | s1872508 |
| Date: | August 31, 2021 |
| Specialisation: | Artificial Intelligence |
| 1st supervisor: | Dr. Frank W. Takes |
| 2nd supervisor: | Dr. Mark P. J. van der Loo |


Master's Thesis in Computer Science


Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

**Abstract**

When sharing sensitive data, it should be made sure that entities represented in it are sufficiently anonymous in order to avoid a possible breach of privacy. In the field of statistical disclosure control, this concept is well studied. However, thus far the majority of work in this field focuses on microdata and (aggregated) tabular data. In this work, we discuss a new measure for anonymity in networks: $d$-$k$-anonymity. It improves upon existing measures (which are in most cases too weak, too strict, or not able to account for triangles) by being parametrized in strictness and taking into account all information in the $d$-neighbourhood of a vertex. This enables the user to select the right level of anonymity based on how much a possible attacker knows. We present an algorithm that can efficiently measure the anonymity and apply it to three well-known graph models with up to 10,000 vertices, as well as a real-world network; the full family network of the Netherlands, consisting of 15.7 million vertices. In our experiments, we find that for graph models most anonymity is lost when measuring 2-$k$-anonymity, and vertices quickly all become unique as the edge density increases. For the family network, over 2.7 million vertices have an anonymity of 1 when measuring 5-$k$-anonymity, implying that they are uniquely identifiable when their exact position in their 5-neighbourhood is known.

# Contents

# Chapter 1

# Introduction

Researching large quantities of data can result in a lot of interesting insights in various domains. An often occurring problem with a lot of data, however, is the possible breach of privacy. Especially when data contains sensitive information on people, this data should not become publicly available. Statistics Netherlands (CBS) is an example of an institution that has access to a lot of data about the Dutch population and works together with other institutes. Before sharing this data, they need to make sure the data shared is properly *anonymized*, for which the original data should often be altered so that the risk of exposing entities represented in this data is sufficiently low. As a result, when a possible attacker tries to de-anonymize this data, this should not lead to a breach of privacy. This is extensively researched in the field of statistical disclosure control, which at present focuses on microdata and (aggregated) tabular data [1, 2].

Recently, CBS has also worked on researching various types of *connections* between people of the Dutch population, which includes connections about family, work, school, households and place of residency. A part of this data is illustrated in Figure 1.1. This data can lead to interesting insights [3] and can be represented by a *network* or *graph* where the vertices are people, and edges represent a type of connection. Sharing this type of data poses new challenges due to the possible disclosure of information originating from the *structural position* of vertices.

**Figure 1.1:** The second largest component of the family network of the Netherlands. Each vertex represents a person, each edge a parent-child relation

Various approaches for anonymizing graphs have been introduced, which are often based on methods in statistical disclosure control. One category of approaches, which will be considered in this thesis and is frequently used on microdata, is *k-anonymity*. This category aims at the vertices such that a graph is said to satisfy $k$-anonymity if each vertex is equivalent to at least $k-1$ other vertices, under some definition of equivalence. Which definition of equivalence is used, should depend on how much information is assumed to be retrievable by a possible attacker. For example, if only the number of direct neighbours is retrievable, this would be sufficient as a measure. Vertices are then considered equivalent, and thus equally anonymous, if they have the same degree. This is an example of a relatively weak measure and often more information than the number of direct neighbours can be obtained. Another extreme is $k$-automorphism [4], where vertices are only equivalent if they are theoretically indistinguishable due to symmetry in the graph structure. For real-world networks, especially large ones, however, this anonymity is hard to achieve since this much symmetry rarely occurs.

When anonymizing in order to obtain anonymity, the graph will be altered and so-called *data utility* will be lost. How much data utility is lost, is dependent on the structure of the graph itself and the anonymization technique, because a more efficient technique might achieve the anonymity with fewer alterations, or might be able to preserve more of the graph properties. In general, however, one could expect that more alterations are required if the measure is very strict than when the measure is weak. Therefore, it is important that when one chooses a measure, one can select the right strictness and therefore a right balance between *anonymity* and *data utility*. For this, an anonymity measure that is parametrized in strictness could be beneficial.

However, there are at least two shortcomings in the currently introduced literature. First, most measures introduced thus far have a fixed level of strictness and are in most cases either too weak or too strict, as we will argue in Section 3.2. Second, most of the literature on the topic of network anonymization focuses on the anonymization process itself, rather than *measuring* anonymity, i.e., measuring how anonymous vertices are in the original unaltered graph. This measurement on its own, however, gives valuable insights into how many vertices are sufficiently anonymous, and how much of the graph should be altered to achieve the desired anonymity.

In order to address these problems, we will discuss a new measure for $k$-anonymity: $d$-$k$-anonymity [5]. This is a measure that is adaptable in strictness with a parameter $d$. When $d = 1$, this measure behaves the same as 1-neighbourhood isomorphism [6], and if the value of $d$ is large enough, it behaves the same as $k$-automorphism [4]. In all other cases, this measure approximates $k$-automorphism. Using this equivalence measure, vertices are equivalent if they have the same structural position in their respective $d$-neighbourhoods.

Additionally, in order to obtain more insights into how the observed anonymity changes as a result of various graph properties and to better understand the observed anonymity in real-world networks, we test this measure on three common graph models: Erdős Rényi [7], Barabási Albert [8], and Powerlaw Cluster graphs [9]. Moreover, we will test the measure on a real-world network, being the family network of the Netherlands, consisting of the parent-child relations of 15.7 million people [3]. CBS allowed temporary secure access to a version of this graph. To summarize, the main contributions of this thesis are as follows:

- Position a new parametrized measure in the existing literature
- Propose a naive and iterative algorithm as well as heuristics for efficiently computing this measure on large, possibly directed, graphs
- Compute and investigate the anonymity of vertices in three well-known graph models
- Compute the anonymity of a large real-world network consisting of 15.7 million vertices
- Investigate the relationship between various graph properties and the anonymity distribution, and the performance of computing this distribution

The remainder of this thesis is structured as follows. First, we will more elaborately summarize the related work done in this field in Chapter 2. Then in Chapter 3, we will go over some background knowledge with regards to graph theory, and anonymity measures. Additionally, the third part of this chapter will briefly describe the three used graph models. Next, Chapter 4 will describe the measure for anonymity that is used in this thesis and how to compute this for a given graph. Chapter 5 contains a more elaborate description of the data used in the experiments. The results of the experiments, both on

graph models and real-world data, can be found in Chapter 6. Lastly, Chapter 7 will conclude this thesis by summarizing the most important findings and discussing some suggestions for future work.

# Chapter 2

# Related Work

In the field of network anonymity, there are at least five important lines of research to be distinguished of which an elaborate summary is given in [10]. This chapter will give a brief overview of three of them, which are most important for this thesis. This does not include measuring data utility and de-anonymization techniques.

**Anonymity measures.** For anonymizing graphs, various measures for anonymity can be chosen. First, one can aim at the vertices, which is the case for *k-anonymity* [4, 6, 11, 12, 13, 14]. Here the graph is said to satisfy $k$-anonymity if for each vertex there are at least $k$-1 equivalent vertices with respect to the used measure. This is also the type of approach considered in this thesis and is often used for microdata, where a value of $k = 3$ is commonly used.

The measures introduced can be categorized in three approaches: degree-based [11, 12], where the number of direct neighbours are taken into account, isomorphism-based [6], where vertices are equivalent if they appear in the same neighbourhood, and automorphism-based [4, 13], where vertices are equivalent if they have the exact same structural position in the graph. Using this last measure, vertices are equivalent if they are theoretically indistinguishable in the graph.

Second, one could also focus on preserving edge privacy [15] when the existence of relations may not be retrieved from this data. Research is also done about privacy in clusters [12], where vertices are divided into clusters, which form supernodes that are then connected to each other.

Additionally, some other approaches extend these measures in order to protect graphs with *vertex labels* [16, 17] or graphs with *labelled edges* [18].

**Anonymization techniques.** Most anonymization techniques use the *edge editing* technique. Using this technique, the graph is altered by adding or modifying edges in the graph in order to increase anonymity. Edges can be added or redirected to a different vertex randomly [19], or by using a more clever approach such as *random walks* [20]. When using random walks, the algorithm starts at a vertex, then follows a number of edges, and adds an edge from the starting vertex to the vertex where the walk ended. More complex approaches have been introduced as well, such as [21], which uses a genetic algorithm or [22] which uses link prediction in order to add edges that are *similar* to the edges already contained in the graph. By using one of these more sophisticated approaches for anonymization, more data utility could be preserved in the anonymized graphs.

Another approach that is introduced, is differential privacy [23, 24]. Using differential privacy, users can ask queries about the structure of the graph. The answers are possibly altered in such a way that no sensitive data can be inferred about the data and sufficient privacy can be guaranteed for the entities represented. This is a common approach in statistical disclosure control as well.

**Measuring anonymity.** Several papers consider graph properties and their relation to anonymity. The work in [12] researches the anonymity distribution of various real-world networks and discusses some theoretical results on ER graphs using a parametrized degree-based measure. The aim of [25] is to measure anonymity in various types of random graphs, similar to what is done in this thesis, and focuses on the 1-neighbourhood of vertices; vertices are equivalent if their 1-neighbourhoods are isomorphic. The work in [26] researches the connection between the de-anonymizability of a graph, and which graph properties

are preserved after anonymization. One of the conclusions is that preserving the degree could be very revealing. In [27], the connection between graph size and privacy is researched by using the linkage covariance metric. This work found that vertices are more anonymous in larger graphs.

This thesis will focus on the *measuring* of anonymity in graphs and discuss the results found on both well-known graph models and real-world network data. In the next chapter, we will discuss various measures of $k$-anonymity in more detail and argue that the measures introduced in the literature thus far are often too weak, or too strict for realistic scenarios in real-world networks. Therefore, this thesis will look into a new measure for $k$-anonymity: $d$-$k$-anonymity. Using this measure for equivalence, vertices are equivalent if they have the same structural position in their respective $d$-neighbourhoods. This measure differs from already existing work as it takes into account all information in the $d$-neighbourhood and is *parametrized* which allows the user to select the desired level of anonymity based on how much an attacker knows. It can be categorized as a *tunable automorphism-based anonymity measure*. Because of this, $d$-$k$-anonymity could be a valuable addition to the already existing measures for $k$-anonymity.

# Chapter 3

# Background

This chapter consists of three parts. The first section will give a brief introduction to graph theory by summarizing some definitions and introducing the notation used throughout this thesis. The next section will focus on measures for anonymity in graphs; here we will discuss the terminology regarding anonymity used in this thesis and a few measures of $k$-anonymity, leading up to a motivation of why $d$-$k$-anonymity [5] is a valuable addition to already existing measures. In the last section, three well-known graph models are briefly discussed; Erdős–Rényi, Barábasi Albert and Powerlaw Cluster Graphs. These graphs will also be used during the experiments in Section 6.

## 3.1 Graph Theory and Notation

A graph (or network) can represent connections between different entities, for example between people. From this, we can obtain various insights in, for example, how people form connections or are connected. A small example is illustrated in Figure 3.1. Here the circles represent **vertices** and the lines **edges**, which are each a connection between two entities.

For this thesis, we will define a graph $G = (V, E)$ as a set of vertices $V = \{1, 2, \ldots, |V|\}$, and edges $E$, which are ordered sets of two vertices $(u, v)$ with $u, v \in V$. For undirected graphs, if $(i, j) \in E$, then $(j, i) \in E$. We will let $|V|$ denote the number of vertices in the graph, and $|E|$ the number of edges.

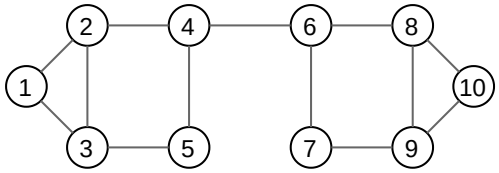



**Figure 3.1:** Example of a graph with 10 vertices, represented by circles, and 13 edges, represented by lines

Furthermore, we can define the degree of a vertex as the number of direct neighbours. In the image above, the degree of vertices 1, 5, 7 and 10 equals 2, and for the other vertices it is equal to 3.

- $degree(v) = |\{(v, w)\ s.t. (v, w) \in E\}|$

In some cases, it can be useful to take the *direction* of edges into account. In the family network studied in this thesis for example, where edges represent a parent-child relation, edge direction can indicate which vertex plays the role of parent or child. The directionality of the edge thus adds extra information. If the edges in a graph are directed, the graph is called a **directed graph**. We distinguish between **ingoing** and **outgoing** edges, and **indegree** and **outdegree**.

- $outdegree(v) = |\{v\ s.t. (v, w) \in E\}|$
- $indegree(w) = |\{w\ s.t. (v, w) \in E\}|$

This thesis will mainly focus on undirected graphs. Unless stated otherwise, it can be assumed that we are discussing undirected graphs.

Additionally, we can define $distance(u, v)$ for two vertices $u, v$ as the minimum number of edges to traverse to get from one vertex to another. Using this definition, the distance from a vertex to itself equals 0. The largest distance to get from a vertex $u$ to every other vertex in the graph is called the **eccentricity** of the vertex, and the largest eccentricity in the graph is the **diameter** of the graph. If the diameter equals 1, the graph is a complete graph and it takes only one edge to get from each vertex in the graph to every other vertex. If this value is larger, for example 5, at most 5 edges have to be traversed to get from every vertex to every other vertex in the graph.

However, sometimes not all vertices can be reached. In this case, the graph consists of multiple **components**, which are not connected to each other. We say that the distance between two vertices, which are each in a different component, equals $\infty$. Therefore, when computing the diameter of a graph, we look only at the distances in the largest component. This component will be referred to as the **giant component**.

The last definition we will introduce is the **induced subgraph**. When given a graph $G = (V, E)$, we can get an induced subgraph by selecting a set of vertices $V' \subseteq V$ and adding all edges from $E$ between these vertices.

**Definition 1** (Induced subgraph)**.** *Given a graph $G = (V, E)$ and a subset of vertices $V' \subseteq V$. The subgraph induced by $V'$ is defined as $G' = (V', E')$ where $E' \subseteq E$ such that for all $v, w \in V'$ it holds that if $(v, w) \in E$ then $(v, w) \in E'$.*

## 3.2 Measuring Anonymity

To arrive at a measure for anonymity, we will first introduce the terminology used for defining equivalency and anonymity. When measuring anonymity, the aim is to create a partition of the vertices, which will be referred to as the **equivalence partition**. Each vertex in the graph will be partitioned in exactly one class: an **equivalence class**. In such a class, all vertices are *equivalent* under the used measure of equivalence. An example of such a measure is the *degree* of a vertex; this measure is also used in [11] and will be used as a first example below.

If a vertex is contained in a class of size $k$, we call this vertex **$k$-anonymous**; the size of the class corresponds to the **anonymity** of the vertices contained in it. Here a higher number (larger class size) indicates higher anonymity for all vertices in this class. If the size of an equivalence class equals 1, the vertex is unique, and if the size equals $|V|$, i.e., all vertices in the graph are in the same class, then all vertices are equivalent with respect to the used measure and thus all vertices are equally and maximally anonymous. The latter is for example the case in a complete graph when using any measure for structural equivalence. Similarly, we call a graph **$k$-anonymous** if all equivalence classes in the equivalence partition have a size of at least $k$. This implies that all vertices are at least $k$-anonymous.

Using this framework for measuring anonymity, various measures for equivalence can be used. We will now discuss in detail three categories of equivalence measures: degree-based, isomorphism-based and automorphism-based.

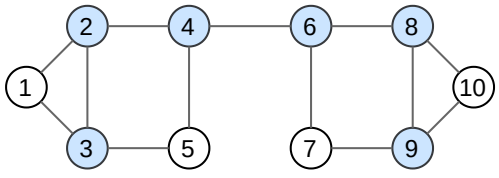



**Figure 3.2:** Graph with vertices coloured according to their degree: vertices with the same colour are in the same equivalence class. Blue vertices have degree 3, white vertices have degree 2

**Degree-based.** An example where vertices are partitioned according to their degree, is shown in Figure 3.2. In this figure, equivalent vertices with the same degree have been partitioned into the same equivalence class, which is shown by the colour (white or blue). Here only two classes are distinguished; vertices with degree 3 (six blue vertices) and vertices with degree 2 (four white vertices). The blue vertices are in this case *6-anonymous*, while the white vertices are *4-anonymous*, making the graph itself *4-anonymous*.

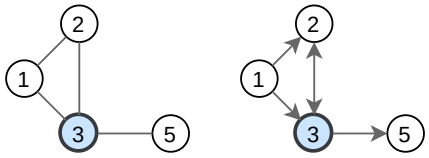
While these equivalent vertices can not be distinguished if only the degree is known, they could still be distinguished when more information about the graph is obtained. For example when comparing vertex 3 and 6, we see that only vertex 3 is part of a triangle, and 6 forms a bridge with 4 to the left part, while 3 has no connection to the other part. Therefore, if a possible attacker can obtain this information, these vertices can still be distinguished. That is why in most cases, one may want to use a more strict measure.

In [12], a tunable degree-based measure is introduced. When computing this measure, first vertices are split based on their degree, then on the degree distribution of vertices at up to distance 1, then distance 2 and so on, until it reaches the desired distance. While this measure is more strict than only the degree of a vertex, it does not take into account triangles, or the exact structure in the neighbourhood, which might be obtainable by a possible attacker.

In the remainders of this section, we will discuss two other measures: 1-neighbourhood isomorphism and $k$-automorphism, measures that do take triangles into account.

**Isomorphism-based.** One example of a more strict measure is one that looks at the **1-neighbourhood** of a vertex, instead of only the degree. This 1-neighbourhood is the *subgraph*, induced over the set of vertices connected to the vertex, and the vertex itself. This is also referred to as the *ego network* of a vertex.

To illustrate this, the 1-neighbourhood of vertex 3, from Figure 3.2, is shown in Figure 3.3a and a directed variant in Figure 3.3b. Note that for the directed variant the directionality of edges is not taken into account when determining the 1-neighbourhood. This subgraph thus contains the vertex itself, all direct neighbours of the vertex, and all edges between this set of vertices. We use this definition for the 1-neighbourhood since it is, in some scenarios, likely that a possible attacker can obtain information about both the ingoing and outgoing edges in the neighbourhood of a vertex. When measuring anonymity, we therefore want to take into account all vertices at a distance of at most $d$ edges, regardless of the directionality of an edge.




**(a)** Undirected **(b)** Directed

**Figure 3.3:** 1-neighbourhood of vertex 3 in the undirected and directed graph. For directed graphs, the directionality of edges is not taken into account when defining the $d$-neighbourhood of a vertex

This leads us to the definition of the more general $d$-neighbourhood as below, which captures all vertices at a distance of at most $d$ from the target vertex.

**Definition 2** ($d$-neighbourhood)**.** *Given a graph $G = (V, E)$ and a target vertex $v \in V$, the $d$-neighbourhood of $v$ is the subgraph $G' = (V', E')$ induced by $V' \subseteq V$ containing all vertices $w \in V$ such that distance$(v, w) \leq d$.*

To compare two 1-neighbourhoods, or two $d$-neighbourhoods in general, we could check if they are **isomorphic**. This isomorphism-based measure is also used to assess anonymity in [6]. Informally, two (sub)graphs are isomorphic when they can be put on top of each other such that all vertices and edges coincide. More formally, two (sub)graphs are isomorphic if there exists a function that maps vertices from one graph to the other that is on-to-one, and invertible. In this function, all vertices that are connected in the original graph, are also connected in the mapped graph.

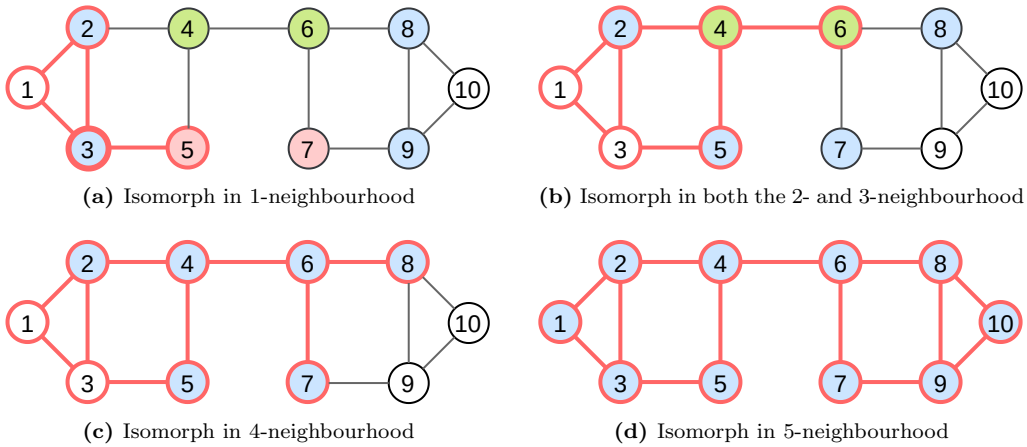
**Definition 3** (Graph Isomorphism)**.** *Given two graphs $G = (V, E)$ and $G' = (V', E')$, the graphs are called isomorphic if there is a bijective function $\phi : V \to V'$ such that for each $u, v \in V$ it holds that $(\phi(u), \phi(v)) \in E'$ if $(u, v) \in E$.*

The 1-neighbourhoods of two vertices are isomorphic if their degrees are equal, and the connections between the direct neighbours are the same. The second requirement makes this measure stronger than

the previous category of measures since the degree-based measure is not able to take the *triangles* in a neighbourhood of a vertex into account. These triangles are formed when two direct neighbours of a vertex are connected, which is also illustrated in Figure 3.3 by vertices 1, 2 and 3. The formation of triangles in graphs is also referred to as **clustering**.

An example where vertices are coloured if they are equivalent with respect to 1-neighbourhood isomorphism is shown in Figure 3.4a. Here equivalent vertices (with respect to the used measure) have the same colour, and to illustrate the $d$-neighbourhoods of vertex 3, vertices and edges in this neighbourhood have a red outline.

In Figure 3.4a, four classes are distinguished, each indicated by a different vertex colour. However, if more information about the graph is known, some of these equivalent vertices can still be distinguished, for example vertices 3 and 8. While they both have 4 vertices in their 1-neighbourhood, 8 will have one more vertex in the 2-neighbourhood than vertex 3.




**(a)** Isomorph in 1-neighbourhood

**(b)** Isomorph in both the 2- and 3-neighbourhood

**(c)** Isomorph in 4-neighbourhood

**(d)** Isomorph in 5-neighbourhood

**Figure 3.4:** Equivalence classes with respect to $d$-neighbourhood isomorphism, with $d \in \{1, 2, 3, 4, 5\}$. Vertices and edges are coloured red to illustrate they are in the $d$-neighbourhood of vertex 3

Alternatively, a measure could check if 2-neighbourhoods of vertices are isomorphic in order to determine equivalence. However, when simply using a larger neighbourhood, some vertices that did not have isomorphic 1-neighbourhoods can have isomorphic 2-neighbourhoods: for example vertices 1 and 3 in Figure 3.4b.

Even worse, when looking at Figure 3.4c, the 4-neighbourhoods of 2, 4, 5 and 6, 7, 8 each contain all vertices of this graph, and the neighbourhood can not grow any larger. This is precisely the case when $d$ is larger than or equal to the eccentricity of the vertex, which equals 4 for all blue vertices, and 5 for the white vertices in Figure 3.4c. If $d$ is larger than or equal to the largest eccentricity of the graph, i.e., larger than or equal to the diameter of the graph, all $d$-neighbourhoods are equal to the entire component they appear in, and thus all vertices in the same component are isomorphic. This is the case in Figure 3.4d, where all vertices are equivalent with respect to 5-neighbourhood isomorphism. Therefore defining vertices as equivalent if their $d$-neighbourhoods are isomorphic can in a lot of cases be insufficient. Furthermore, vertices can be equivalent in a larger neighbourhood while they are non-equivalent when taking into account a smaller neighbourhood. Structural properties at a smaller distance, such as degree, are not taken into account when looking at larger neighbourhoods.

**Automorphism-based.** The last measure for equivalency that we will discuss, is **$k$-automorphism** as introduced in [4]. An automorphism is similar to an isomorphism, but it is a mapping from a graph *onto itself*. In [4], a graph is $k$-anonymous if there are at least $k$ automorphic functions such that each vertex is mapped onto a different vertex for each of these automorphic functions.

If a vertex $v \in V$ is mapped to a vertex $v' \in V$ by at least one existing automorphic function, we say that they are in the same **orbit**. Using the notion of $k$-automorphism, the orbit of a vertex also corresponds to the equivalence class of a vertex. Thus, for a graph to be $k$-anonymous with respect to automorphism, the orbits should all have a size of at least $k$. An example where vertices in the same equivalence class are coloured accordingly, is shown in Figure 3.5.

**Definition 4** (Graph Automorphism)**.** *Given a graph $G = (V, E)$, an automorphic function $\gamma$ is defined as the bijection $\gamma : V \to V$ such that for each $u, v \in V$ it holds that $(\gamma(u), \gamma(v)) \in E$ if $(u, v) \in E$.*

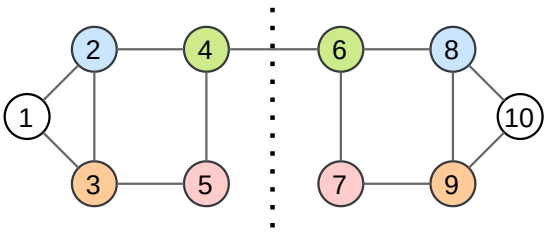



**Figure 3.5:** Equivalence classes with respect to automorphism

In the example illustrated in Figure 3.5, all vertices are 2-anonymous with respect to $k$-automorphism, and can not be further distinguished based on their structural position. This shows the strictness of this measure; theoretically, vertices in the same equivalence class, or orbit, are indistinguishable even when all possible information about the graph is known. A disadvantage of this approach is that this implies that there should be symmetry in the graph in order for vertices not to be unique. For a graph to be $k$-anonymous with respect to automorphism, the graph should be symmetric in at least $k - 1$ points. In the example, it is symmetric in the bridge between vertices 4 and 6, making it 2-anonymous. But for real-world graphs, especially large ones, it is very unlikely that the graph is symmetric. Due to this, $k$-automorphism is often a too strict measure in practice, while the other two measures looking at degree or the 1-neighbourhood, are in most cases too weak.

In the upcoming chapter, we will therefore discuss a new measure for anonymity which, by using a parameter, can differ in strictness.

## 3.3 Graph Models

In the upcoming subsections, we will briefly describe the three graph models used for the conducted experiments. These descriptions include how the graphs are generated and can be used to help understand the emergence of various of their properties.

### 3.3.1 Erdős Rényi

Erdős Rényi (ER) graphs [7] are the simplest random graphs to test graph algorithms on. Using this model, one can easily differ both the size of the graph and its density. A small example of an ER graph can be found in Figure 3.6.

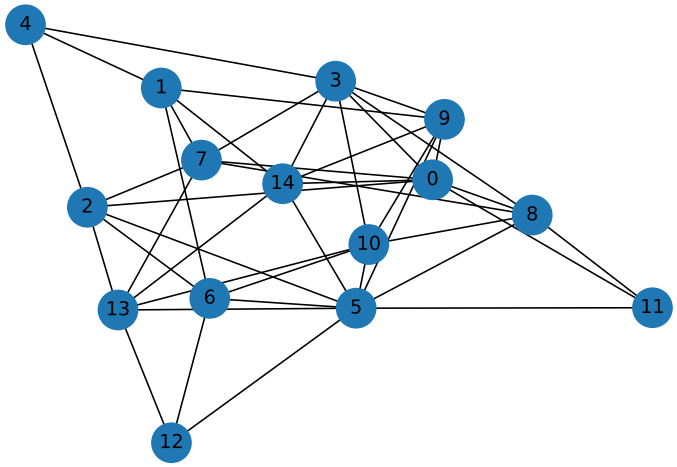



**Figure 3.6:** Example of ER graph with $|V| = 15$ and $p = 0.3$

Given the desired number of vertices $|V|$ and a probability $p$, the procedure to generate an ER graph is as follows:

1. Iterate over all possible edges the graph can contain, which equals $|V|\frac{|V|-1}{2}$ for undirected graphs
2. Add each edge with a probability $p$

Using the parameter $p$, the density of the graph can be controlled. A value of $p = 0.0$ will result in a graph without edges, whereas $p = 1.0$ will result in a complete graph with all $|V|\frac{|V|-1}{2}$ edges. Edges are added uniformly at random, which implies there is no bias in which edges are added and which are not. This results in a Poisson-shaped degree distribution. As a result, when $0.0 < p < 1.0$, the expected degree of a vertex equals $p \times (|V|-1)$, but the exact degree of a vertex, or the total number of edges, is not set.

### 3.3.2 Barabási Albert

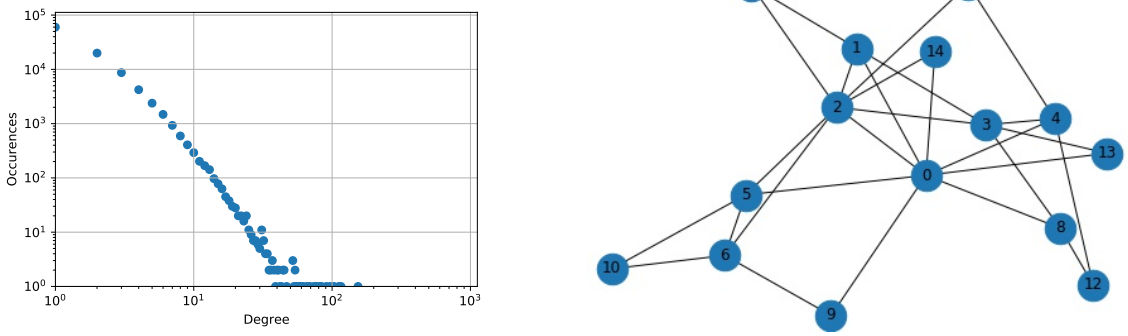
The Barábasi Albert model introduced in [8] aims to recreate the power-law degree distribution that is often observed in real-world networks. This makes this model more realistic than the ER model and is achieved by taking into account two observations that often occur in real-world graphs: growth and preferential attachment.

Growth refers to how a graph expands when vertices are added; new edges are created when new vertices are added to the graph. The second mechanism, preferential attachment, means that vertices are more inclined to connect to vertices with a high degree, than vertices with a low degree. This is different from being equally likely to connect to any vertex, as was the case for ER graphs. The probability $p_{attach}$ that an edge is attached to a given vertex $v \in V$ is computed using the formula below. Note that vertices that are not added to the graph yet, with a degree of 0, are not taken into account and have a probability of 0.0 to be chosen.

$$p_{attach}(v) = \frac{degree(v)}{|E| * 2}$$

An example of a BA graph can be found in Figure 3.7b, and an illustration of the power-law degree distribution in Figure 3.7a. For this thesis, BA graphs are generated as follows:

1. Start with $m$ vertices, where the first $(m)$ vertices form a complete graph [1]
2. Add the remaining $|V| - m$ vertices, by adding $m$ edges for each vertex. Here each vertex has a probability of $p_{attach}$ to be chosen.




**(a)** Example of BA degree distribution with $|V| = 10.000$ and $m = 1$

**(b)** BA graph with $|V| = 15$ and $m = 2$

**Figure 3.7:** BA graph illustration of degree distribution and an example graph

[1] This starting condition can differ based on the implementation used. For BA graphs this is igraph [28] for and for HK graphs networkx [29]

### 3.3.3 Powerlaw Cluster Graph

In [9], the powerlaw cluster graph model was introduced by Holme and Kim (HK). This is an extension of the Barábasi Albert model that preserves the property of the power-law degree distribution, but adds one step to the graph generation, which makes it possible to add more clustering (i.e. triangles) to the graph. This is in order to better mimic the structure of real-world graphs where high clustering is often observed.

This extra step adds a triangle to the graph, if this is possible, and is executed with a probability of $p_{triangle}$. An example of an HK graph is illustrated in Figure 3.8. HK graphs are in our case generated as follows:

1. Create a star-shaped graph by connecting $m$ vertices with a center vertex, being the center of the star
2. Add the remaining $|V| - m - 1$ vertices, by adding $m$ edges for each
3. For the first edge, select a target vertex, where each vertex in the graph has a probability of $p_{attach}$ to be chosen. For the remaining $m - 1$ edges:
   (a) With probability $p_{triangle}$ create a triangle. This is done by randomly selecting a neighbour of the target vertex, such that the source vertex is not yet connected to this neighbour. Then add an edge between these two vertices.
   (b) Else, or if the triangle generation failed, select a vertex via the preferential attachment mechanism

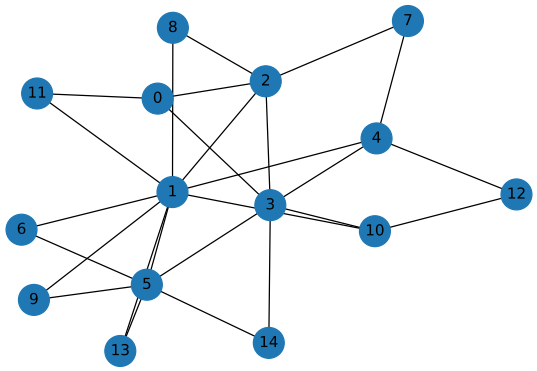



**Figure 3.8:** Example of HK graph with $|V| = 15$, $m = 2$ and $p_{triangle} = 0.05$

# Chapter 4

# Approach

The previous chapter has discussed measures for determining if two vertices are equivalent. This measure of equivalence is required to compute the equivalence partition of a graph, and therefore the anonymity of its vertices. It started with the degree of vertices, which is the weakest measure discussed. Then it looked at isomorphisms of $d$-neighbourhoods, and ended with automorphisms, which in theory is a perfect definition for anonymity, since using this measure equivalent vertices will be theoretically indistinguishable; even if all information about the graph is known. This last measure is, however, often too strict in practice, since it demands symmetry in the graph.

In the next section, we will define the $d$-$k$-anonymity measure. The sections thereafter will explain how this measure is computed and discuss heuristics used to speed up the computation.

## 4.1 A Definition for d-k-Anonymity

Below we discuss a new measure for determining if two vertices are equivalent which we will refer to as $d$-$k$-anonymity [5]. For two vertices to be equivalent under this measure of equivalence, they should have the same structural position in their respective $d$-neighbourhoods. These $d$-neighbourhoods should also be isomorphic. This measure for $d$-$k$-anonymity can also be used for directed graphs and graphs with labelled vertices and edges.

**Definition 5** ($d$-$k$-anonymity)**.** *Given a graph $G = (V, E)$ and a distance $d > 0$, two vertices $v_1, v_2 \in V$ are equivalent with respect to $d$-$k$-anonymity if the following two properties hold:*

- *At least one isomorphism between the $d$-neighbourhoods of $v_1$ and $v_2$ exists*
- *There is at least one such isomorphism in which $v_1$ is mapped onto $v_2$*

The measure has a parameter $d$ which can be used to determine the strictness of the measure. It has the property that in order for two vertices to be equivalent with respect to $(d+1)$-$k$-anonymity, they will also have to be equivalent with respect to $d$-$k$-anonymity, for each value of $d$. The proof is contained in [5].

If $d = 1$ it behaves the same as 1-neighbourhood isomorphism defined in [6] and explained in Figure 3.4a. If $d$ is larger or equal to the diameter of the graph, it behaves the same as $k$-automorphism, discussed in [4] and illustrated in Figure 3.5. As discussed in Section 3.2 and illustrated in Figure 3.4d, the neighbourhood considered for each vertex, when $d \geq diameter$, consists of the component the vertex appears in and will therefore be the same for each vertex contained in it. As a result, the neighbourhoods of all vertices in this component will be isomorphic. The second requirement for equivalency is that there should be at least one isomorphism mapping the vertices onto each other, which is, in this case, the same as an automorphic function and implies vertices should be in the same orbit in order to be equivalent. Therefore, in this case, the measure is the same as $k$-automorphism.

For other values of $d$, it looks at a $d$-neighbourhood that does not contain the entire component; in these cases, the measure approximates $k$-automorphism and checks if the vertices have the same structural position in their $d$-neighbourhood.

As stated at the beginning of Section 3.2, we say a vertex is $d$-$k$-anonymous if it is in an equivalence class, with respect to $d$-$k$-anonymity, of size at least $k$. Equivalently, a graph is $d$-$k$-anonymous if all equivalence classes in the graph have a size of at least $k$, and thus each vertex is at least $d$-$k$-anonymous.

## 4.2 Computing d-k-Anonymity

Given a graph $G = (V, E)$ and a distance $d$, we want to determine the *equivalence partition* of the vertices with respect to $d$-$k$-anonymity. Here the set of all vertices is partitioned in equivalence classes which each contain one or more vertices $v \in V$. The most obvious method, to which we will refer to as the *naive algorithm*, is to just start with the class of all vertices and group them based on the $d$-$k$-anonymity property. Here each vertex is compared to a vertex of an equivalence class until it finds the equivalence class it belongs in. If the vertex does not fit in an already existing equivalence class, a new class is created. Determining if two vertices are equivalent consists of two parts, corresponding to the two requirements in Definition 5.

First, to determine if two $d$-neighbourhoods are isomorphic, we can use the notion of canonical labelling [30] as defined below. This function transforms the graph into its canonical form such that all isomorphic permutations of the graph will have the same canonical form. From this, it follows that two graphs are isomorphic if their canonical labellings are equal.

Consequently, once we have the canonical form of the $d$-neighbourhoods, we can simply check if they are equal. If this is the case, the neighbourhoods are isomorphic, otherwise, they are not.

**Definition 6** (Canonical Labelling)**.** *A canonical labelling is a function $\mathcal{C}$ that transforms the graph into its canonical form such that for two given graphs $G = (V, E)$ and $G' = (V', E')$, any automorphic function $\gamma$ and graph $G^\gamma$ to which this function $\gamma$ is applied, the following two properties holds:*

- $\mathcal{C}(G^\gamma) = \mathcal{C}(G)$.
- $\mathcal{C}(G) = \mathcal{C}(G')$ *if $G$ is isomorphic to $G'$.*

Second, we need to determine if there is an isomorphism for which the vertices are mapped onto each other. This can be done by using the orbits of the $d$-neighbourhoods. The canonical labelling gives an isomorphism that maps vertices from one graph onto the other graph. If the first vertex is mapped onto a vertex that is in the same orbit as the second vertex, there is at least one isomorphism that maps the first vertex onto the second vertex, thus satisfying the second requirement.

This naive approach is summarized in Algorithm 1.

**Algorithm 1** NAIVE-D-K-ANONYMITY

```
1:  Input: Graph G = (V, E), equivalence class eq = V, distance d
2:  eq_partition_new = {}
3:  if d ≤ 0 then                                   ▷ Distance 0, no information is known
4:      return {eq}
5:  end if
6:  for v1 ∈ eq do                                  ▷ For each vertex, find or create new eq
7:      sub1 = get_neighbourhood(G, v1, d)
8:      cansub1 = C(sub1)
9:      is_new_class = false
10:     for eq2 ∈ eq_partition_new do               ▷ Check if vertex v1 fits in already existing class
11:         v2 = v ∈ eq2                            ▷ Get a vertex from eq2
12:         sub2 = get_neighbourhood(G, v2, d)
13:         cansub2 = C(sub2)
14:         if are_same(cansub1, cansub2, v1, v2) then   ▷ Check two equivalence criteria
15:             eq2 = eq2 ∪ v1
16:             is_new_class = true
17:             break
18:         end if
19:     end for
20:     if ! is_new_class then                      ▷ Vertex gets new class
21:         eq_partition_new = eq_partition_new ∪ {v1}
22:     end if
23: end for
24: return eq_partition_new
```

As input of Algorithm 1, a graph and an equivalence class to start with are required. For this naive approach, the latter corresponds to the set of all vertices. First, in line 3, we check if the distance is larger than 0. If this is not the case, no information can be used so all vertices can be viewed as equivalent and the algorithm is done. The for loop in lines 6–23 iterates over all vertices in the equivalence class given as input, and partitions each in the correct class.

In order to find the right equivalence class for a vertex, we iterate over all equivalence classes (lines 10–19) from which we each select a vertex (line 11). Since all vertices in the same equivalence class are equivalent, this vertex can be selected randomly. This is because if a given vertex is equivalent to this vertex, it is equivalent to all vertices in the class.

In order to test for equivalence, we first compute the canonical labelling of the $d$-neighbourhood of both vertices (lines 7, 8 and 12, 13 ). Then, we check for equivalence which consists of two parts. First, it is checked if the canonical labellings of the two neighbourhood graphs are identical (the first requirement in Definition 5). Second, we check if $v_1$ is mapped onto a vertex that is in the same orbit as $v_2$ (the second requirement in Definition 5). If these two requirements hold for the two vertices, we add the new vertex to this class and continue with the next vertex (lines 14–17). Otherwise, the algorithm continues the loop by comparing $v_1$ to a vertex in the next equivalence class. Once a vertex has been compared to all existing equivalence classes and still is not partitioned in a class, a new equivalence class is created containing only this vertex (lines 20–21). When every vertex has been partitioned into an equivalence class, this algorithm is finished and the equivalence partition is generated.

## 4.3 Iterative Computation of d-k-Anonymity

Using the approach in Algorithm 1, each vertex is in the worst case compared to all other already existing equivalence classes, resulting in $\mathcal{O}(|V|^2)$ required comparisons in the worst case. These comparisons (equivalence checks) can be relatively expensive since this approach immediately looks at neighbourhoods with radius $d$.

A way to counter this effect is to start at a neighbourhood of distance 1, and iteratively increase this distance by one, until this equals the chosen distance $d$. We will refer to this as the *iterative approach*. This uses the property that in order for two vertices to be equivalent with respect to $(d+1)$-$k$-anonymity, they have to be equivalent with respect to $d$-$k$-anonymity.

In Algorithm 2 is summarized how this iterative splitting of the equivalence classes can be done. It starts with one equivalence class containing all vertices in the graph. The inner loop of the algorithm (lines 4–7) iterates over the obtained equivalence classes and aims to split them into new classes using new information of a (possibly) larger neighbourhood. This process is the same as the process described in Algorithm 1, but a smaller equivalence class, not consisting of all vertices in the graph, is given as input eq after the first iteration.

**Algorithm 2** ITERATIVE-D-K-ANONYMITY

```
1:  Input: Graph G = (V, E), distance d
2:  eq_partition = { {0, 1, ..., |V|} }          ▷ Equivalence partition contains 1 class
3:  for i = 1 to d do
4:      for eq in eq_partition do                 ▷ Iterate over classes, split them further
5:          eq_new = NAIVE-D-K-ANONYMITY(G, eq, i)   ▷ Call function in Algorithm 1
6:          eq_partition_new = eq_partition_new ∪ eq_new   ▷ Insert all new classes
7:      end for
8:      eq_partition = eq_partition_new
9:      eq_partition_new = ∅
10: end for
11: return eq_partition
```

After splitting the equivalence class, it inserts all the obtained classes into the new equivalence partition (line 8). Each iteration of the outer for loop, the algorithm increases the neighbourhood distance by 1, until it reaches distance $d$.

## 4.4 Heuristics

Two different heuristics have been used to speed up the process of partitioning vertices into equivalence classes with respect to $d$-$k$-anonymity. In Algorithm 1, to place a vertex in the correct class, it iterates over all newly generated classes (line 10) until it finds the correct equivalence class. With the help of heuristics, we can find a smaller set of candidate vertices, which results in fewer required comparisons to find the correct equivalence class for a given vertex, while still generating the correct results. Essentially this means that on line 10 of Algorithm 1 a function *candidates(eq_partition_new, v)* is defined where *candidates(eq_partition_new, v)* $\subseteq$ *eq_partition_new*. This can be particularly useful in situations where the size of eq_partition_new is very large (which more often occurs during the early iterations, thus for smaller distances), or when the comparisons are expensive (for larger distances).

Using one of the heuristics, if no existing equivalence class has the same property, we can group the vertex into a new equivalence class. If there is at least one class with this property, the vertex will have to be compared to vertices in those classes, until it finds the correct class.

The first requirement for two vertices to be equivalent with respect to $d$-$k$-anonymity, is that their respective $d$-neighbourhoods should be isomorphic. Because of this requirement, we know that $d$-neighbourhoods should have some similar properties which do not change under isomorphism, which are also called *graph invariants*. Graph invariants that we used as heuristics are:

- Vertices and edges: the number of vertices and edges in the $d$-neighbourhood
- Degree distribution: which equals the degree distribution in the $d$-neighbourhood (for directed graphs, the in- or out-degree can be chosen as well)

The first heuristic is the simplest, and will overall be inexpensive to compute. The second heuristic could be more expensive depending on the number of different degrees in the $d$-neighbourhood. When this number is larger, this heuristic will be more expensive, but also more informative. Note that this distribution also contains information about the vertices and edges in the graph; the number of vertices can be computed by counting the number of degrees taken into account, and the number of edges by computing the sum of all degrees multiplied by their respective occurrence.

# Chapter 5

# Data

In this chapter, we will discuss in more detail the data used, consisting of the three graph models, Erdős Rényi (ER), Barabási Albert (BA) and the Powerlaw Custer Graph (HK), and the family network of the Netherlands.

## 5.1 Graph Models

For graph models, one can control the edge density of the graph by using a parameter. In [7], the addition of more edges to the ER graph, while the number of vertices stays constant, is referred to as *the evolution of random graphs*. In this section, we aim to better understand these graph models by studying different graph properties and how they change during the evolution of each graph model. Overall, real-world networks are very sparse and thus show most similarity to graphs early in this evolution. Therefore, we mainly focus on the early stage of the evolution where vertices have a relatively low average degree.

During this section, we will consider graphs with 100 vertices, and an average degree ranging from $[0, 1, \ldots, 99]$ for ER, and for BA and HK $m = [1, 2, \ldots, 99]$. This shows the entire evolution of each graph model. For each combination of graph model and average degree (ER) and $m$ (BA and HK), we generated 20 graphs using random seed 456, and report the average of the found results. For the scatter plots, all results are presented, and thus sum to $20 \times |V|$ per column.

### 5.1.1 Erdős Rényi

Subsection 3.3.1 described that edges in this graph are assigned randomly, therefore ER graphs are also referred to as random graphs. Through this, there is no bias in which edges to add to the graph and which not. In [7], five different phases are distinguished in their evolution. Due to the random nature of ER graphs, we say that the properties described are almost certain to consist in any generated random graph, with a probability close to 1.0. The five phases are as follows:

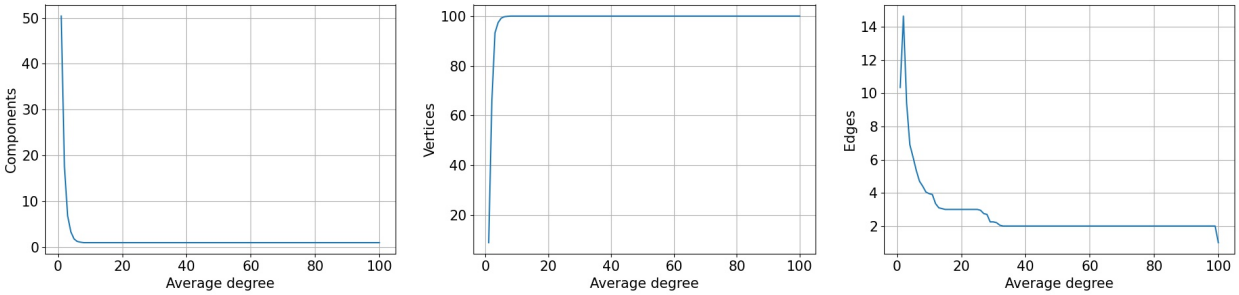
1. The graph consists of only trees
2. The first cycles are formed
3. Structure changes abruptly; small components start to melt into the giant component with a complex structure. Other components are relatively small, mostly trees
4. All vertices almost surely become connected
5. The graph becomes asymptotically regular; the graph's structure starts to become the same until it is fully connected

Additionally [12] theoretically researches the anonymity in ER graphs when using the degree-based anonymity measure as discussed in Section 3.2. In this work, three phases of ER graphs are distinguished:

1. Sparse graphs: average degree $c$ where $c > 1$ is a constant and $|V| \to \infty$. Vertices can not be distinguished. This is caused by the sparseness that does not allow the graph to create diversity in structure. For graphs of less than $10^6$ vertices, some re-identification occurs

2. Dense graphs: average degree $c \times \log |V|$. For sufficiently large graphs, vertices can not be identified at distance 1 for $c > 0$. At distance 2, however, all nodes are identified when $c > 1$

3. Super dense graphs: average degree $\frac{|V|-1}{2}$. When the graph contains half of the edges, the probability for two vertices to be equivalent at distance 3 is very small $2^{-cn}$ with $c > 0$




**(a)** Number components **(b)** Largest component size **(c)** Diameter

**Figure 5.1:** Global properties in ER graphs

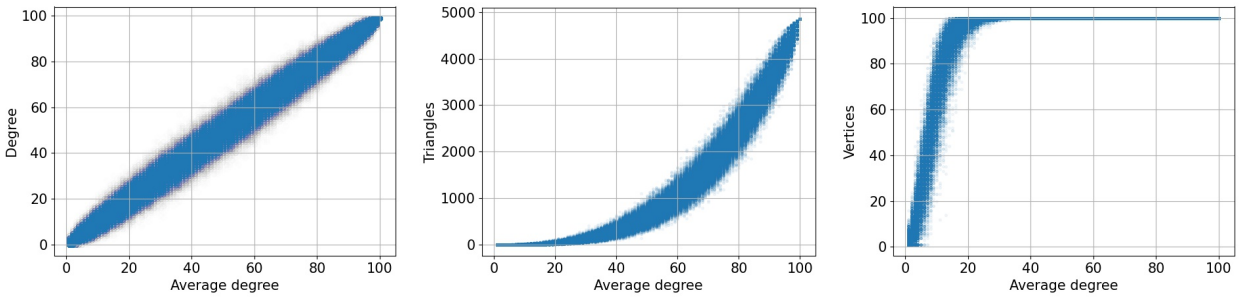
We now turn to the structural properties of ER graphs. The first three Figures 5.1a, 5.1b and 5.1c contain descriptive results concerning global graph properties: the number of components, largest component size and diameter. These results can be explained by the five phases found in [7]. First, the graph is has no edges and during phase 1 it will generate small trees. Next, during phase 2, the first few cycles will be formed. After that, phase 3 takes place where the structure changes abruptly as most vertices will be consumed by the giant component.

The first three phases can be observed by the start in all figures: the largest (giant) component grows to contain all vertices during the start, and we also see that the size of the largest component increases quickly from 0 to almost 100 during the range of average degree 0–5. In this same range, we also see the number of components decrease from 100 to almost 1. These results also show that the remaining components that were not yet in the giant component were relatively small, containing only a few vertices each.

The increase in diameter shows how the longest shortest path between two vertices quickly becomes larger. Since most vertices are new in the giant component, it could be expected that they are not connected to many other vertices in the graph yet, which could explain the larger diameter observed. Additionally, Figure 5.2b illustrates that more clustering occurs at a larger average degree, which might indicate that before this the giant component is less dense.

Thereafter, phase 4 takes place where the giant component starts to absorb the remaining vertices of the graph. In Figures 5.1a and **??**, this is shown at the part where the increase (largest component size) and decrease (the number of components) declines. After this, both values stay constant at value 100 (largest component size) and 1 (the number of components). The diameter starts to decrease very quickly during this phase, which indicates that the giant component becomes denser. The first four phases thus take place between average degrees of approximately 0–7. Therefore, the last phase, phase 5, takes place during the largest part, until average degree 99. During this phase, we see that the diameter converges to 2.0, which means that the 2-neighbourhood of each vertex will contain all vertices in its component. This implies that 2-$k$-anonymity will already look at the entire component, behaving the same as $k$-automorphism. This is also shown in Figure 5.2c.

The last three results, which focus on local graph properties, can be found in Figures 5.2a, 5.2b and 5.2c. In the left-most Figure 5.2a, we see a linear relationship between the average degree and degrees of vertices. Here the degree can differ per vertex; at most parts, a difference of at most 20 is observed. At the beginning and end of the evolution, this difference is smaller since it starts at an average degree of 0 (no edges) and in the end, converges to 99 (complete graph). Note that the degree distribution also corresponds to the size of the 1-neighbourhood of a vertex, for which is therefore not a lot of variation.




**(a)** Degree distribution **(b)** Triangles per vertex **(c)** 2-neighbourhood sizes

**Figure 5.2:** Local properties in ER graphs

Figure 5.2b shows how many triangles per vertex are observed. At the start, until average degree 20 approximately, this equals almost 0 for all vertices. After this point, it starts to increase, resulting in a more clustered structure, until the end when the graph becomes fully connected.

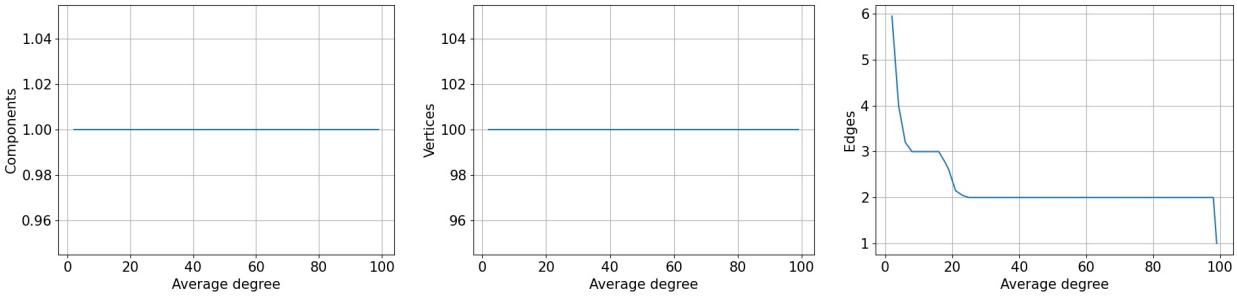
The right-most Figure 5.2c illustrates how the sizes of the 2-neighbourhood change. This figure shows a similar pattern to that of the largest neighbourhood sizes in Figure 5.1b, which implies that overall the 2-neighbourhood of each vertex contains a large fraction of the component it belongs in. In the beginning, we see a very large increase, and after an average degree of 20, it contains almost all vertices in the graph. After average degree 30, the 2-neighbourhood always contains all vertices.

### 5.1.2 Barabási Albert

Barabási Albert graphs are generated with in mind the notions of preferential attachment and growth in networks. This makes BA graphs more similar to real-world networks than ER graphs, while this model still enables one to differ in size and density of the graph. Due to the difference in the generation process, the evolution of BA graphs differs from that of ER graphs.

When looking at the evolution of largest component sizes and number of components in Figures 5.3a and 5.3b, we see that the giant component immediately contains all vertices in the graph. Therefore the graph is immediately connected, which differs from the ER graphs where this happened after an average degree of 5 approximately.

In Figure 5.3c, we see the change of the diameter in BA graphs. Since these graphs consist of only one component at the beginning, and then only becomes denser, the diameter starts at its peak and thereafter immediately decreases. Thereafter, the diameter quickly decreases and reaches a value of 3 at average degree 15, and a diameter equal to 2, at an average degree of 23. Compared to ER graphs, the peak appears earlier, the diameter decreases faster and reaches a value of 2 earlier.




**(a)** Number components **(b)** Largest component size **(c)** Diameter

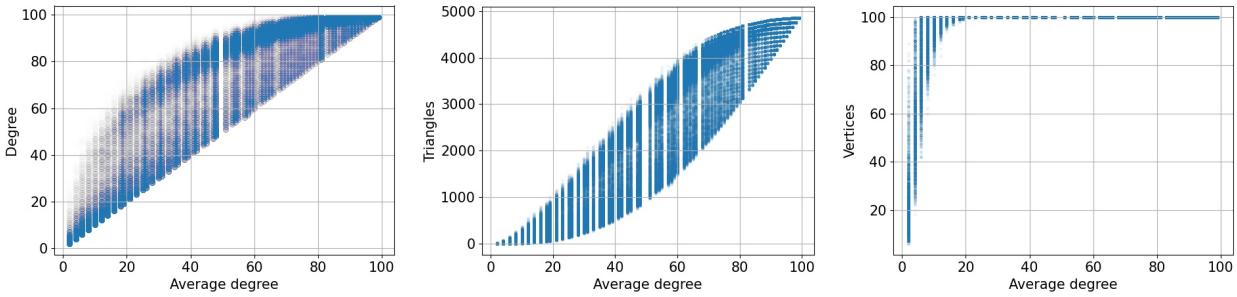
**Figure 5.3:** Global properties in BA graphs

When we look at the local graph properties, we see in Figure 5.4a that the degree distribution for BA graphs is much more diverse than for ER graphs. The lowest degree follows a linear pattern since each

vertex has at least $m$ edges. The difference between the highest and lowest degree does first increase as the average degree increases, and after the top around average degree 40, the degree distribution starts to converge until all vertices have an equal degree when the graph is fully connected.

For the number of triangles per vertex in Figure 5.4b, we observe different results than for ER graphs. The minimum number of triangles does seem to follow a similar trend as for ER graphs, but for BA graphs the diversity in the number of triangles per vertex is larger.

Figure 5.4c shows the size of the 2-neighbourhood of the graph. This follows a similar pattern as for ER graphs, and around the same average degree of 30, the 2-neighbourhood of each vertex consist of all vertices.



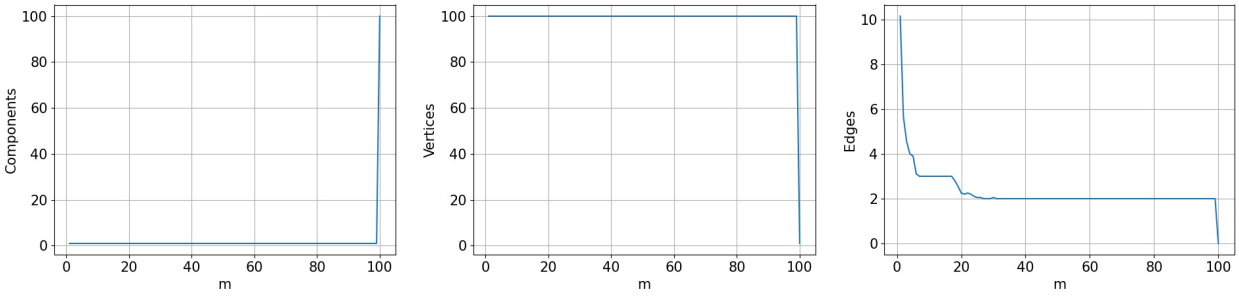

**(a)** Degree distribution **(b)** Triangles per vertex **(c)** 2-neighbourhood sizes

**Figure 5.4:** Local properties in BA graphs

### 5.1.3 Powerlaw Cluster Graph

The Powerlaw Cluster graph has been introduced in [9] by Holme and Kim (HK). This model extends the BA model with a parameter that enables one to add more clustering. However, the implementation of the initial generation step for BA graphs differs per framework, as discussed in Section 3.3. Therefore, for this part, we will discuss HK graphs with the horizontal axis labelled with $m$; the number of edges to attach from each new vertex.

When taking into account global graph properties, the behaviour of HK graphs and BA graphs are the same: they both consist of only one component immediately and the change in diameter of both models also show a very similar pattern.

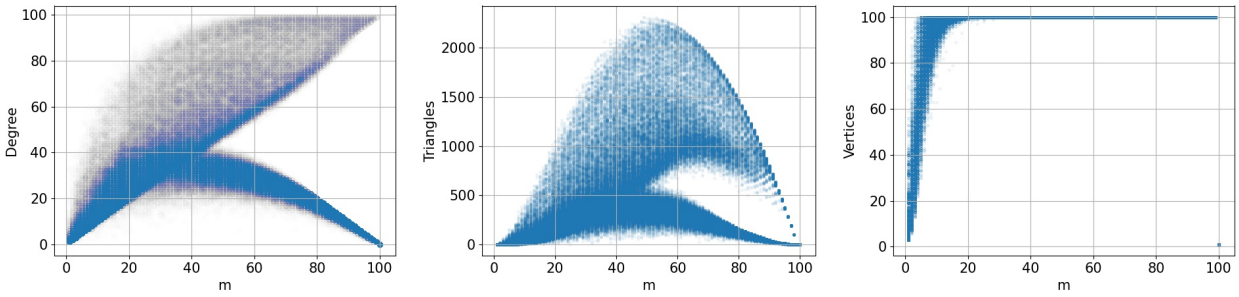



**(a)** Number components **(b)** Largest component size **(c)** Diameter

**Figure 5.5:** Global properties in HK graphs

For the local graph properties, some large differences can be found, which are likely caused by the difference of generating the graphs, and the additional clustering step. In Figure 5.6a, the top half of the degree distribution is quite similar to that of BA graphs, but less dense. The bottom of the figure, however, shows that a lot of vertices have a degree smaller than $m$. This is caused by the difference in generating the two graphs; while BA graphs start with a fully connected graph consisting of $m$ vertices, the HK graphs start with a star graph, where the first $m$ vertices start with a degree of 1. These last

vertices are not always guaranteed to obtain a degree of $m$. At $m$= 99, this is a very extreme case: first 99 vertices are added, to which the $100^{th}$ vertex is attached. This results in a star graph where all vertices have a degree of 1, except the $100^{th}$ vertex, which has a degree of 99. When generating this graph, the clustering step is not used.




**(a)** Degree distribution **(b)** Triangles per vertex **(c)** 2-neighbourhood sizes

**Figure 5.6:** Local properties in HK graphs

For the number of triangles per vertex in Figure 5.6b, we also see a large difference between BA graphs. Some vertices have a very large number of triangles; this first increases from $m$ = 0–50, and thereafter decreases. While during the first half fewer edges are added from each vertex, this process, including the clustering step, also happens for more vertices which causes the number of triangles per vertices to increase. However, after $m$ = 50, more vertices are part of the star graph HK graphs start with, and the number of vertices from which $m$ edges have to be attached, and possible triangles are added, decreases.

During this first half, the number of triangles in HK is higher than for BA graphs. The part that we will focus on while discussing the results (average degree 1–100, for graphs with 1,000 and 10,000 vertices) also falls in this first half.

When comparing the sizes of 2-neighbourhoods in Figure 5.6c in HK to BA graphs, we see a similar pattern: the 2-neighbourhood sizes first increases very fast, until at some point it always contains all vertices. The latter takes place after $m$ = 20, while for BA graphs this takes place at an average degree of 30 approximately. The increase of 2-neighbourhood size also happens slightly faster for HK graphs, and the differences found in size are smaller for this graph.

## 5.2 The Family Network of the Netherlands

The real-world network used for the experiments is the family network of the Netherlands. CBS allowed temporary secure access to an unlabelled version of this graph. The network is derived from relational data of the entire population of the Netherlands and we will consider full parent-child relations (i.e., parents are the biological parents of the child). Due to that this data was stored in an edge list, vertices without edges (people with no (living) parent or children) do not occur in the network data. People who were not alive when this data was derived are also not included in the family network. To understand this network in more detail, this section will discuss some of its properties including diameter, number of components, and degree distribution.

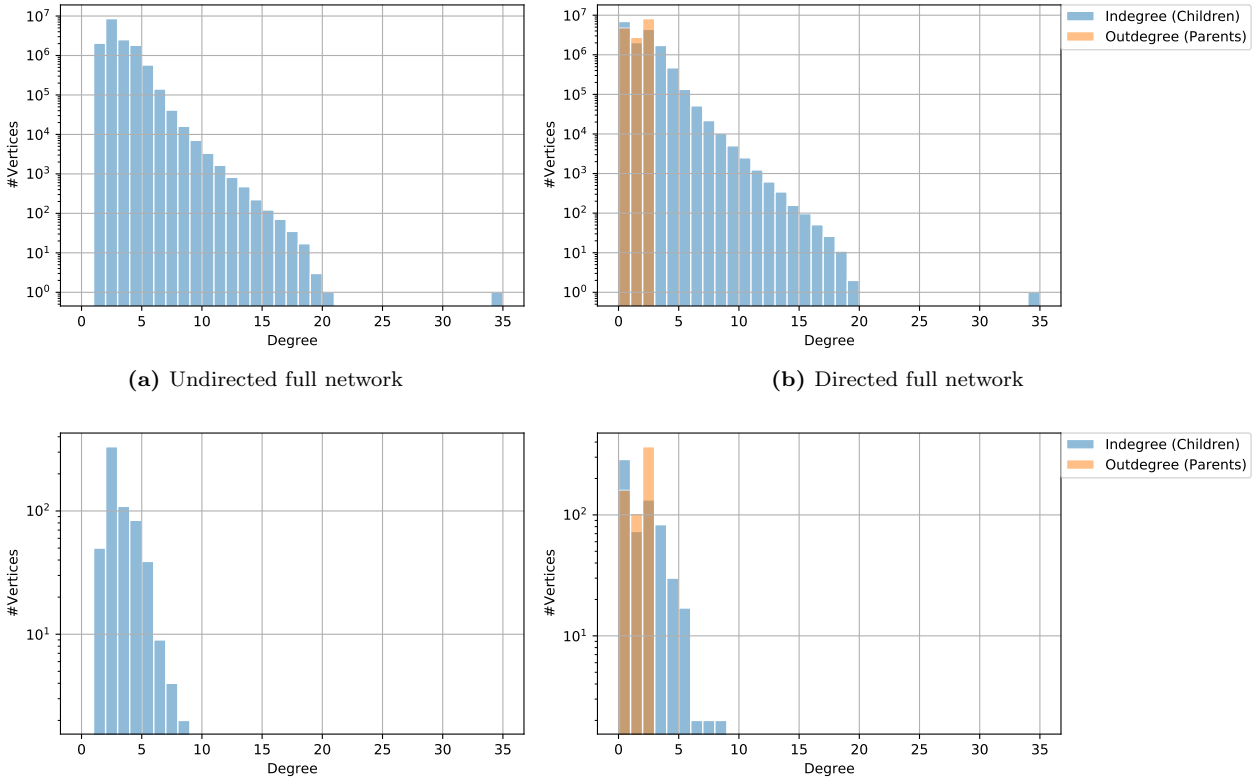
In Table 5.1 some of the properties of the full family network and the second largest component can be found. The full network contains 15.7 million vertices approximately, containing a large fraction of the Dutch population, which was approximately 17.2 million at the time this dataset was derived. While the largest component in the family network contains a very large fraction of the vertices (over 9 million vertices), the second largest component is significantly smaller (629 vertices). The diameters in each of these components, however, are very large: 235 for the largest component, and 63 for the second largest component. This can also be explained by the type of data; in full parent-child relations, not a lot of connectedness is expected which might result in a larger diameter.

| | Family network | Second largest component |
|---|---|---|
| Vertices | 15,760,721 | 629 |
| Edges | 19,160,768 | 835 |
| Average degree | 2.23 | 2.66 |
| Components weakly connected | 963,754 | 1 |
| Largest component size | 9,123,881 | 629 |
| Diameter | 235 | 63 |

**Table 5.1:** Properties of the full family network and the second largest component

In this network, vertices have a small degree on average. This can be partially explained by the type of edges, which represent full parent-child relations. A person can have at most 2 parents, and one can not have an unlimited number of children. Moreover, most households in the Netherlands have only a few children; 1 or 2 are the most common. [1] This can also be seen in the degree distributions of the full family network and the second largest component in Figure 5.7.

The degree distributions for both the full family network and the second largest component can be found in Figure 5.7. The indegree equals the number of children, and the outdegree the number of parents. The latter can either be 0, 1 or 2. For the outdegree (number of parents) we see that having either no or both parents occurs slightly more often than having one parent for both the full network, and the second largest component. For the full family network, we see that 0-5 children are the most common, and after that, the number of occurrences decreases by a factor ten. Some very large degrees occur as well, up to 20 and one vertex has a degree of 35. Vertices with a high degree are very rare compared to vertices with lower degrees.




**(a)** Undirected full network

**(b)** Directed full network

**(c)** Undirected second largest component

**(d)** Directed second largest component

**Figure 5.7:** Degree distribution in Family network, second largest component

[1] www.cbs.nl/nl-nl/cijfers/detail/71487ned

In the second largest component, we see a less diverse distribution. Here we see more clearly that most people tend to have a lower degree, i.e., a smaller household and a smaller number of children. Having 0 or 2 children is the most common, the rest is not observed more than 100 times. A larger number of children, larger than 5, is very uncommon in this component and no vertices with a degree higher than 10 are contained in this component.

# Chapter 6

# Experiments

The $d$-$k$-anonymity measure has been implemented in C++ using the nauty framework [30], which can efficiently compute the canonical labelling of graphs, as described in Section 4.2, and obtain the orbits of vertices in the graph. The source code for the implementation of $d$-$k$-anonymity is available.[1] For the experiments, we use the iterative approach and a heuristic for a larger speed-up. The experiments done can be partitioned into three parts.

First, we will discuss the results of experiments performed to measure the anonymity distribution in graph models. During the second part of the experiments, we will discuss the results found in the real-world network: the family network of the Netherlands. The last part will focus on experiments about the runtime and performance of the algorithm measuring $d$-$k$-anonymity. Here we will study the effect of using heuristics, the difference between the iterative and naive algorithms, and the runtime per iteration.

The experiments with graph models are conducted on a machine with Intel Core i7-8700 processor (6 cores, 12 threads), 16 GB RAM, NVidia GeForce GTX 1050 with 2 GB VRAM and 0.5 TB local HDD storage. For the experiments with the family network, we used a machine with 16 Intel Xeon Gold 6146 processors.

## 6.1 Anonymity in Graph Models

For the experiments in this section, we have used graphs with 100, 1,000 and 10,000 vertices and an average degree ranging from 0–100 for ER, and for BA and HK 1–100 and the constant value of $p_{triangle} = 0.1$ for HK graphs. For graphs with 100 vertices, this will show the entire range of possible degrees, but for larger graphs, we focus on more sparse graphs. Each combination is repeated 10 times to take the randomness of the generated graphs into account and using random seed 156.

For these graph models, we will discuss the anonymity distribution of $d$-$k$-anonymity found and how it changes when the average degrees increases. Here we discuss results with distance $d = 1$ up to 5 with the main focus on distances 1 and 2. For each combination of graph type, average degree and distance, is shown which equivalence classes appear. During the discussion of the results, we try to explain the observations by referring to the local and global graph properties found in Section 5.1.

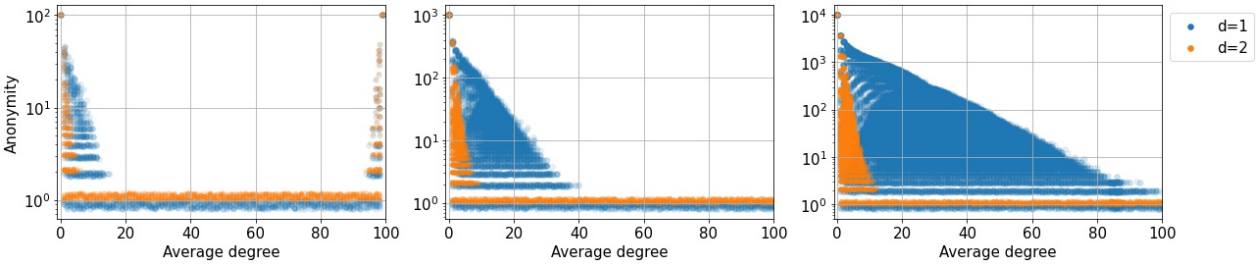
### 6.1.1 Anonymity: Erdős Rényi

First, we will discuss the anonymity distributions in ER graphs. The results for 1- and 2-$k$-anonymity can be found in Figure 6.1. This type of plot will be used throughout the discussion of anonymity in graph models to show their $d$-$k$-anonymity distribution. The horizontal axis denotes the average degree of the graph, which ranges from 0 (no edges) to 100 (full graph). For ER graphs, a step size of 1 is used. On the vertical axis, the *anonymity* is denoted, which equals the size of an equivalence class. Drawing a horizontal line in this type of plot can denote a graph anonymity threshold.

Every dot in the figure denotes the size of an observed equivalence class, which therefore equals the anonymity for the vertices in the class. For example, at the top left in Figure 6.1a, there is a dot at average

[1] https://github.com/RacheldeJong/dkAnonymity

degree 0, anonymity 100, which means there is exactly one equivalence class of size 100. At this point, the graphs have no edges and therefore all vertices are equivalent. After the average degree of 20, most classes have size 1, which implies all vertices are unique when taking into account the 1-neighbourhood in ER graphs with 100 vertices. Note that in order to illustrate the results both at distance 1 and 2, the points with anonymity 1 are non-overlapping, but all have anonymity 1. Furthermore, the results of 10 different graphs are reported for each possible average degree, so for each, the number of dots times anonymity will not sum to $|V|$ but to $10 \times |V|$.




**(a)** $|V| = 100$ **(b)** $|V| = 1,000$ **(c)** $|V| = 10,000$

**Figure 6.1:** Anonymity distribution with respect to 1- and 2-$k$-anonymity for ER graphs with average degree ranging from 0 to 100, and number of vertices 100 (a), 1,000 (b) and 10,000 (c)

For ER graphs with 100 vertices, in Figure 6.1a, we can clearly distinguish 4 phases in the anonymity distribution:

1. When the graph has no edges, all vertices are indistinguishable
2. The anonymity decreases and some vertices are already unique; for distance 1 this phase takes longer than for distance 2
3. All vertices are unique both at distances 1 and 2
4. The anonymity increases when the graph becomes fully connected

This is similar to the phases of ER graphs discussed in [12]; phase 1 corresponds to the sparse graphs, where vertices are indistinguishable. The second phase corresponds to the dense phase where at distance 1 vertices could not be identified for sufficiently large graphs. At distance 2 however, vertices could already be identified if the average degree is larger than $c\ log(|V|)$ with $c > 1$ a constant and $|V| \to \infty$. When $c = 1$, this corresponds to an average degree of 6.6, 10.0 and 13.3 for ER graphs with 100, 1,000 and 10,000 vertices respectively. These relatively small differences in average degree can also be seen in our results. The exact average degree where this occurs does match with our results for ER graphs with 100 and 1,000 vertices (6, 10). For ER graphs with 10,000 vertices, this occurs slightly later than expected at an average degree of 18, which might be as a result of using 10 simulations and reporting all equivalence class sizes found. The last phase, where the graph becomes fully connected, is not discussed in [12]. During this phase, the anonymity increases until all vertices are indistinguishable; the graph is then fully connected.

When looking at distance 1 separately, this second phase takes place until average degree 20, 40 and after 100 for ER graphs with 100, 1,000 and 10,000 vertices respectively. After this, all vertices are unique at every distance. The difference in when this transition takes place can be explained by the difference in graph size; one of the conclusions in [27] tells us that vertices are more anonymous in larger graphs. Additionally, the results found in Figures 5.4a and 5.4b give another possible explanation using local graph properties. When comparing 1-neighbourhoods, we compare degrees and how the direct neighbours are connected (triangles). Earlier we saw that the degree of vertices does not differ greatly for ER graphs with 100 vertices. Hence when we only account for the degree, the vertices should be grouped in a relatively small number of classes ($\approx 20$ for ER graphs with 100 vertices), which is still expected as the number of edges increases resulting. This results in high anonymity when only the degree is taken into account. The number of triangles per vertex, however, did show an increase after average degree 20, which very likely

causes the decrease in anonymity, and increase in uniqueness of vertices after this value.

For the anonymity at distance 2, we again see the decrease in anonymity at the start. For this distance, the vertices become unique at a much smaller average degree; 6, 10, 18 compared to 20, 40, 100 in the previous results. This is similar to the empirical results found in [12] where most anonymity is lost at distance 2. For this distance, the difference in graph size results in a much smaller difference in the point when all vertices become unique compared to distance 1.

In Figure 5.1c, we also saw that the diameter drastically decreases after the average degree of 6, and became 2 after average degree 20. When also taking into account that after average degree of 20 the graph consists of one component (Figure 5.1a), this implies that the 2-neighbourhood of all vertices consists of the entire graph. In those cases, the measure is equal to $k$-automorphism, which, as discussed, is a very strict measure that classifies two vertices as equivalent only if their structural positions in the graph are exactly the same. Before the diameter equals 2, the 2-neighbourhood sizes of vertices are also very close to the sizes of the components in the graph. While the sizes are not yet equal, this does imply that 2-$k$-anonymity is very close to $k$-automorphism already for most ER graphs.

When observing the results for the 3-$k$-anonymity, which can be found in Figure A.1 of Appendix A, some slight difference can be seen compared to 2-$k$-anonymity. When looking at the 4 or 5-neighbourhood, again a bit more anonymity will be lost, but it largely remains the same. For the discussed graph models, most anonymity is lost at distance 2.

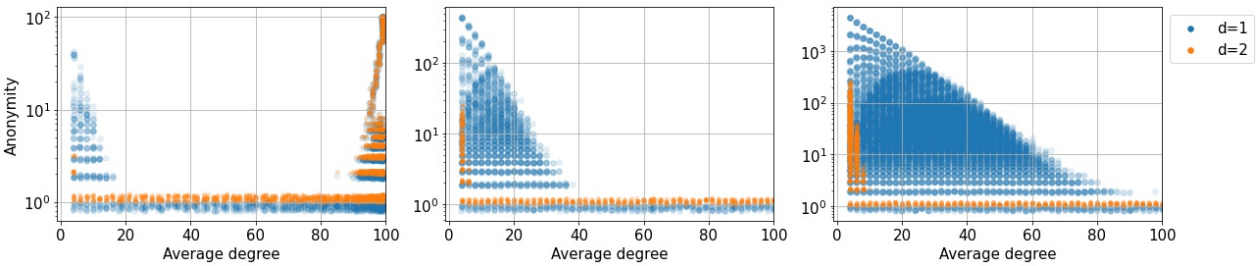
### 6.1.2 Anonymity: Barabási Albert

To discuss the results for BA graphs, we will again first look at results for distances 1 and 2 in Figure 6.2. For this type of graph, we see similar patterns in the anonymity distribution as for ER graphs. Note that due to the difference in generating the graphs, the results seem less dense; while for ER a step size of 1 is used, this can not be done for BA graphs, which uses a parameter $m$, resulting in different step sizes.

The largest difference between the results for ER and BA graphs is that the decrease of anonymity at the start happens faster for BA graphs. Especially at distance 2, all vertices are almost immediately unique. This is likely caused by the observed larger diversity in degree (Figure 5.4a) and triangles per vertex (Figure 5.4b) that were observed for BA graphs in Subsection 5.1.2.

The faster decrease of anonymity at distance 2, can be explained by the finding that BA graphs consist of one component immediately and the diameter of the graph becomes 2 after average degree 20 (in a graph with 100 vertices) already, which means the 2-neighbourhood of each vertex contains the entire graph. Before this, the diameter is larger with a maximum of 12 which results in some equivalent vertices.

At higher average degrees towards 100 in graphs with $|V| = 100$, we see the anonymity increasing in an odd pattern; one line with a much higher anonymity can be distinguished, while there are also a lot of vertices with a lower anonymity. In ER graphs, we also saw the anonymity grow at high average degrees. For BA graphs, this seems to occur as well.




**(a)** $|V| = 100$ **(b)** $|V| = 1,000$ **(c)** $|V| = 10,000$

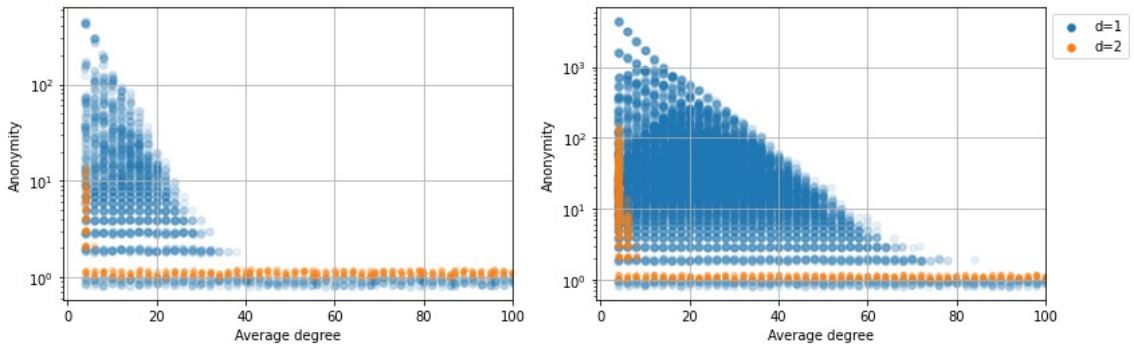
**Figure 6.2:** Anonymity distribution with respect to 1- and 2-$k$-anonymity for BA graphs with average degree ranging from 1 to 100, and number of vertices 100 (a), 1,000 (b) and 10,000 (c)

When observing the results for 3-$k$-anonymity in BA graphs, which can be found in Appendix A Figure A.2, some slight difference in anonymity can be observed compared to 2-$k$-anonymity, which is

similar to our findings in ER graphs. When looking at the 4 or 5-neighbourhood, again a bit more anonymity will be lost, but it largely remains the same. For this graph model, most anonymity is again lost at distance 2.

### 6.1.3 Anonymity: Powerlaw Cluster Graph

For HK graphs the anonymity distributions observed, in Figure 6.3, are very similar to those of BA graphs. The vertices do become unique slightly earlier both at distance 1 and 2; before average degree 40 (1,000 vertices) and around 80 (10,000 vertices) for distance 1 and average degrees 2, 6 for distance 2. This difference could be explained by the additional clustering that is added to the graph, or the different first step in the generation algorithm of the graph.




**(a)** $|V| = 1,000$ **(b)** $|V| = 10,000$

**Figure 6.3:** Anonymity distribution with respect to 1- and 2-$k$-anonymity for HK graphs with average degree ranging from 1 to 100, and number of vertices 1,000 (a) and 10,000 (b)

When looking at larger distances, similar as for ER and BA graphs, a slight change in anonymity is observed, but most of the anonymity is lost at distance 2. Results for HK graphs can be found in Appendix A, Figure A.3.

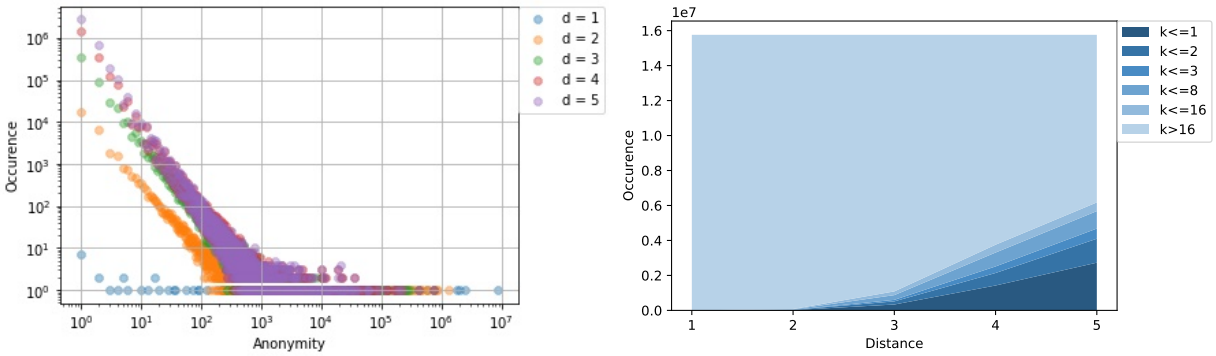
## 6.2 Family network

In this section, we will discuss the results for the family network. We present results of anonymity in both the full family network and the second largest component. For the second largest component, we also discuss results on the directed variant and research the effect it has on the anonymity of vertices. Due to runtime problems with the directed variant (for some neighbourhoods, the algorithm got stuck at computing the canonical labelling), results are discussed with distance 1, 2 and 3. Lastly, in order to get insights in when a vertex is anonymous in this network, and when not, we will look at some of the most and least common neighbourhoods found.

### 6.2.1 Anonymity: Family Network

In Figure 6.4 the anonymity distribution for the family network of the Netherlands can be found. To get a good idea of the anonymity, we present these results in three ways. First, Figure 6.4a shows a scatter plot of how often each equivalence class size occurs for each distance, which shows which class sizes occur. The horizontal axis denotes the anonymity or the equivalence class size, the vertical axis the occurrence such that each dot denotes the number of occurrences of an equivalence classes size.

Second, for a better idea of which fractions of vertices have a low anonymity, Figure 6.4b illustrates how often each equivalence class occurs with a focus on smaller equivalence classes. Since small fractions are not visible in the figure, Table 6.1 shows the exact number of vertices that have a low anonymity at every distance. Together this demonstrates which fraction of vertices can be considered at *risk* of being de-anonymized when this data is shared. Since in statistical disclosure control a value of $k = 3$ is commonly used, vertices that are less than 3-anonymous should be anonymized.




**(a)** Anonymity distribution with logscale

**(b)** Anonymity landscape $k \leq 1$, 2, 3, 8, 16 and $k > 16$

**Figure 6.4:** $d$-$k$-Anonymity in full family network with respect to $d$-$k$-anonymity for $d = \{1, 2, 3, 4, 5\}$

| k | $d$=1 | $d$=2 | $d$=3 | $d$=4 | $d$=5 |
|---|---|---|---|---|---|
| 1 | 7 | 17,383 | 335,143 | 1,423,905 | 2,728,574 |
| 2 | 4 | 12,784 | 179,428 | 714,844 | 1,369,664 |
| 3 | 3 | 5,637 | 87,150 | 374,739 | 581,271 |
| 4 | 4 | 6,071 | 84,660 | 312,008 | 428,872 |
| 5 | 10 | 4,130 | 47,440 | 119,530 | 136,560 |
| $\geq 5$ | 15,760,693 | 15,714,715 | 15,026,900 | 12,815,695 | 10,515,780 |

**Table 6.1:** Number of $\leq k$ anonymous vertices in full family network

In the results, we see that the larger the distance, the less anonymous vertices are. When we look at distance 1, almost all vertices are $> 5$-anonymous and only a negligibly small fraction is less than 5-anonymous. This can be explained by the degree distribution in this network. Most vertices have a low degree, and because the graph does not contain (many) triangles, this results in a lot of similarity in 1-neighbourhoods and an overall high anonymity.

For the graph models, we saw most of the anonymity was lost at distance 2. For the family graph, this at first does not seem to be the case as most of the vertices are still $> 16$-anonymous. However, the exact number of unique vertices does grow over 2,000 times as large when going from distance 1 to distance 2, which is proportionally the largest difference in growth observed.

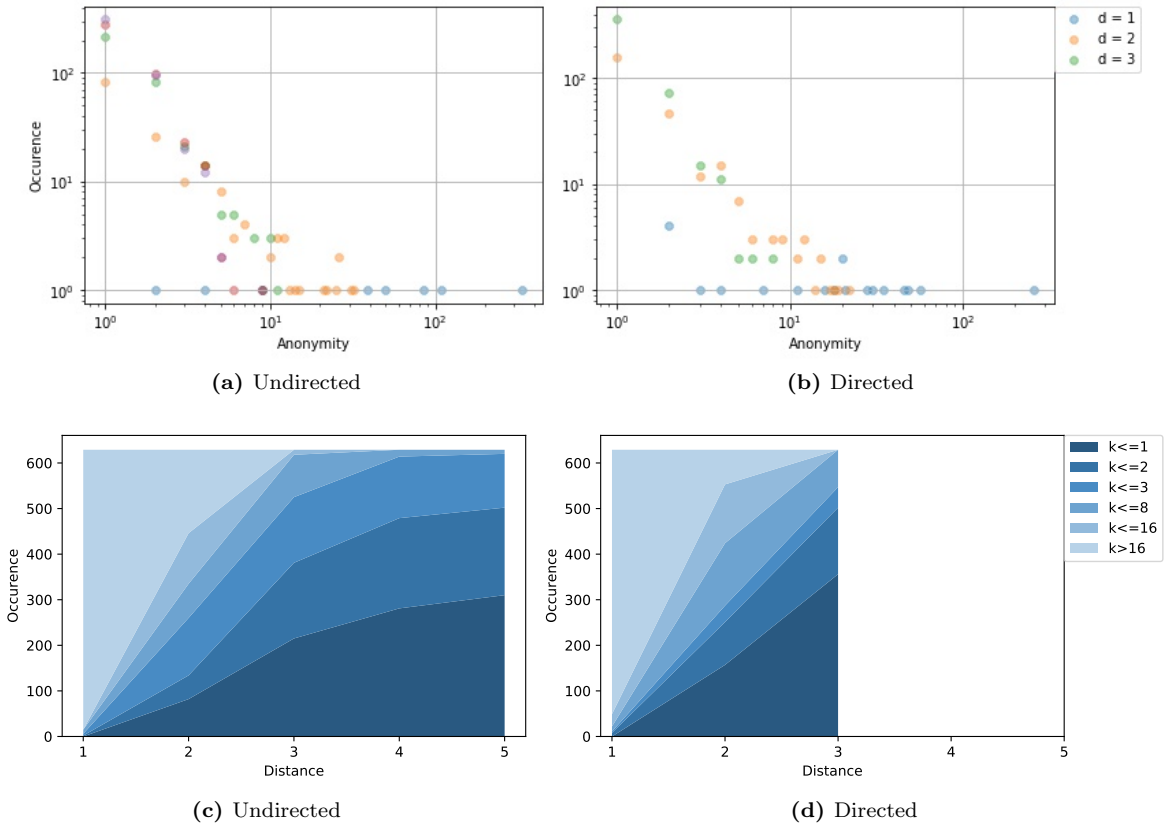
After distance 3, the size of the smaller classes seems to grow linearly; from these vertices, the group of unique vertices is the largest observed. At the largest distance of 5, approximately 4 million vertices are at most 2-anonymous, which shows that more than a quarter of the people represented are not sufficiently anonymous and should be anonymized. Moreover, 2.7 million people can be uniquely identified when an attacker knows their exact 5-neighbourhood.

Since the diameter of the family network (235) is much larger than 5, it could be expected that the anonymity decreases further when the distance increases further to a larger value. For this thesis, however, we will remain at this distance and note that knowing all the family relations of a person until distance 5 is a lot of knowledge and it is unlikely for an attacker to know the exact structural position of a person, i.e., the exact neighbourhood and position up to distance 235.

### 6.2.2 Anonymity: Second Largest Component

In Figure 6.5 the anonymity distributions for the second largest component can be found, both for the undirected and directed variant. Again the scatter plot (a, b) shows which equivalence class sizes are found, and the stack plots (c, d) more clearly illustrate the fraction of vertices with low, or high anonymity.

These figures show that for this component a larger fraction of vertices is not sufficiently anonymous; at distance 2 almost a sixth of the vertices is not sufficiently anonymous, and at distance 5, this is almost five sixth of the vertices. This difference can be explained by the difference in graph size; one of the conclusions of the work in [27] was that more anonymity can be preserved in larger graphs compared to




**(a)** Undirected **(b)** Directed

**(c)** Undirected **(d)** Directed

**Figure 6.5:** Anonymity distribution (a, b) and landscape (c, d) for second largest component of the family network with respect to $d$-$k$-anonymity with $d = \{1, 2, 3, 4, 5\}$. Undirected (a, c) and directed(b, d)

small graphs. This is a result which we also saw for the graph models.

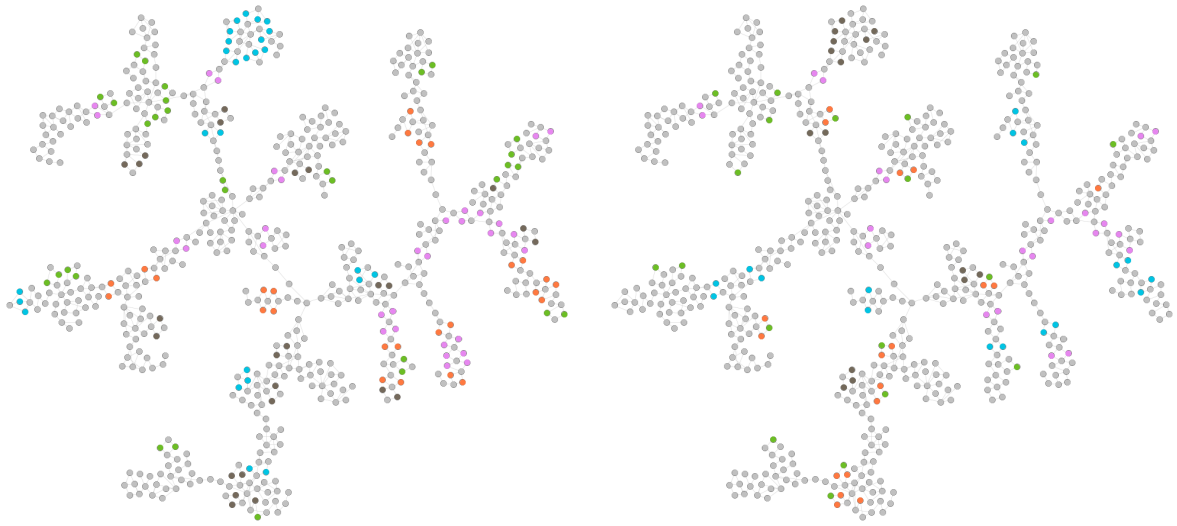
Here we also see the large decrease in anonymity at distance 2; while at distance 1 there are no unique vertices, this number almost equals 100 at distance 2 for the undirected variant, and over 100 for the directed variant. At distance 2, there are also no equivalence classes with a size larger than 100.

In the figures, we can also see the effect of adding directionality to the edges. Figure 6.5d shows that at distance 1 the graph contains approximately 50 vertices that are $\leq 16$-anonymous, while this was approximately 10 vertices for the undirected variant. The anonymity also decreases more quickly when the distance increases; at distance 3 approximately 500 vertices are $\leq 2$-anonymous, while for the undirected variant this equals less than 400 vertices.

For a visual illustration of the anonymity distribution and how it changes due to a larger distance and directionality of edges, Figure 6.6 contains illustrations of the largest component generated with Gephi [31], where vertices are coloured according to the equivalence class they belong in. Only the vertices in the 5 largest equivalence classes, and thus the highest anonymity, are coloured. The remaining vertices are grey. These figures show that at distance 2 a lot of anonymity is lost, both for the directed and undirected variant. When adding directionality to the graph, some anonymity is lost as well, but this difference is not as large as when increasing the distance from 1 to 2.

**(a)** Undirected $d = 1$

**(b)** Directed $d = 1$



**(c)** Undirected $d = 2$

**(d)** Directed $d = 2$

**Figure 6.6:** Illustration of the second largest component with vertices coloured according to their equivalence class. From largest to smallest: **pink**, **green**, **orange**, **blue**, **brown** and the remaining vertices are coloured **grey**. Undirected (a, c) and directed (b, d) distance 1 (a, b) and 2 (c, d)

### 6.2.3 Directionality

To show in more detail the effect of edge directionality on anonymity per vertex, Figure 6.7 shows the ratio directed / undirected class size for each vertex. Here a ratio of 4, for example, means that the equivalence for this vertex in the undirected variant is 4 times as large as for the directed variant.

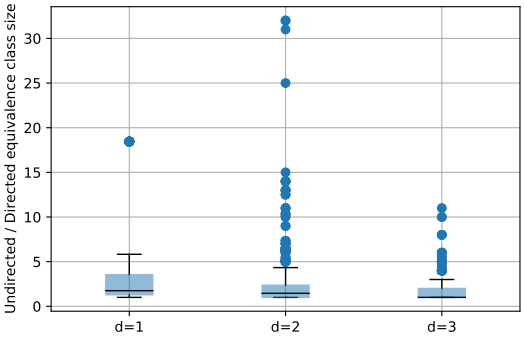



**Figure 6.7:** Anonymity loss when taking into account directionality of edges in the second largest component

In Figure 6.7 we see that at distance 1, the median is the highest, and most vertices lose anonymity when directionality is taken into account. Here there are also fewer outliers than for other distances; only one. At distance 2, the largest losses in anonymity are observed, but the median is slightly lower than for distance 1 meaning that overall, less anonymity is lost due to the directionality of edges than at distance 1. At distance 3, overall less anonymity is lost. There are still a few outliers, but the median equals 1.

Overall, the effect of edge directionality on anonymity can greatly differ per vertex. At distance 1, most vertices lose some anonymity, at distance 2 this difference can be very large for some vertices but also very small to nothing. At distance 3, edge directionality affects fewer vertices.

### 6.2.4 Common and Rare Neighbourhoods

In this subsection, we will look at the most and least frequently occurring neighbourhoods found in the family network. Vertices with an often occurring neighbourhood will have a high anonymity in general, while vertices with a rare neighbourhood will be easier to identify and are possibly unique.

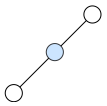
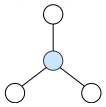
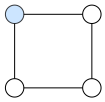
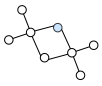
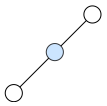
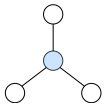
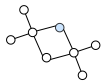
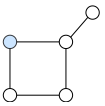
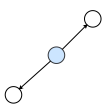
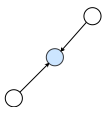
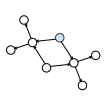
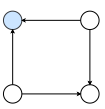
| | $d = 1$ | | $d = 2$ | |
|---|---|---|---|---|
| Full undirected | Equivalence classes: 42 | | Equivalence classes: 35,795 | |
| | 8,593,877 | 2,513,231 | 1,297,365 | 866,167 |
| |  |  |  |  |
| Part undirected | Equivalence classes: 8 | | Equivalence classes: 166 | |
| | 332 | 109 | 32 | 31 |
| |  |  |  |  |
| Part Directed | Equivalence classes: 20 | | Equivalence classes: 259 | |
| | 257 | 57 | 22 | 19 |
| |  |  |  |  |

**Table 6.2:** Most common neighbourhoods with their frequency and the number of equivalence classes in the graph for each distance

Table 6.2 contains the most common neighbourhoods in the family network: both for the full network (undirected) and for the second largest component (undirected and directed). In all neighbourhoods illustrated in this Table, vertices have low degrees ranging from 1 to 4. For the undirected variants, a degree of 2 or 3 is most common. A degree of 2 could mean 2 parents, 1 child 1 parent or 2 children, but due to the missing edge directionality, we can not determine which vertex plays which role. The figures in the table also tell us that vertices with degree 1 are apparently less common.

When looking at the directed variant at distance 1, vertices with degree 2 are the most common; here they either have 2 parents, or 2 children (outgoing edges imply that the vertex is the parent in this relation, ingoing vertices a child). Apparently having either 2 children or 2 parents is more common in this component than having 1 parent and 1 child.

Distance 2 shows larger neighbourhoods where vertices still have a low degree but a much lower occurrence. Here the highest degree observed equals 4 in the 2-neighbourhoods with 8 vertices. For the directed variant, this shows that the vertex has a sibling, two parents and 4 grandparents. For the undirected variant, this neighbourhood is also very common (second most common for the full network, most common for the second largest component). Here, however, it can also denote a different relation. Due to the missing edge directionality, the outer vertices could also be half-siblings, or the inner vertex itself might be the grandparent of the four outer vertices, for example. This is an example of a neighbourhood where edge directionality is required to be certain about the exact role of vertices in the family relation.

The other graphs that occur are square-shaped; these neighbourhoods have 2 parents with 2 children. In the directed variant, the target vertex is the parent of two children, for the undirected variant the target vertex might either be a parent, or a child. For the undirected version of the second largest component, we see a square with one vertex attached. This attached vertex can for example be a grandchild, grandparent or a child from another family.

In Table 6.3 images of the least frequently occurring neighbourhoods can be found. The 1-neighbourhoods all have a very high degree; especially for the full undirected variant where the second 1-neighbourhood contains a degree of 30 and the second one 35. While the directionality is not given, we can conclude from this that at least 28 or 33 of these vertices are children.

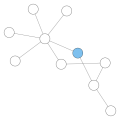
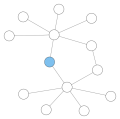
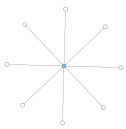
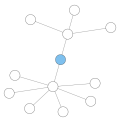
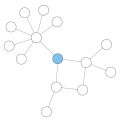
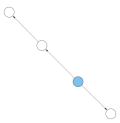
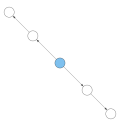
| | $d = 1$ | | $d = 2$ | |
|---|---|---|---|---|
| Full undirected | Equivalence classes: 42 | | Equivalence classes: 35,795 | |
| | 1  | 1  | 1  | 1  |
| Part undirected | Equivalence classes: 8 | | Equivalence classes: 166 | |
| | 2  | 4  | 1  | 1  |
| Part Directed | Equivalence classes: 20 | | Equivalence classes: 259 | |
| | 2  | 2  | 1  | 1  |

**Table 6.3:** Most rare neighbourhoods with their frequency and the number of equivalence classes in the graph for each distance

At distance 2, we see some complex structures for the undirected variants. Here vertices with high degrees occur in a star-like shape, and cycles of length 5 are found as well. These neighbourhoods are all unique. For the directed variant, the 2-neighbourhoods are not very complex. Here a lot of vertices are already unique, which apparently also includes 2-neighbourhoods with a simple structure and low

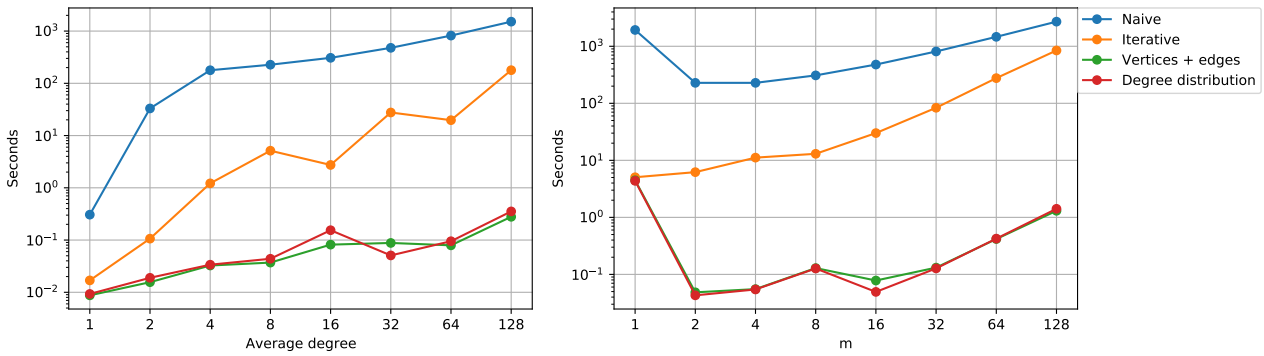
degrees representing a child with two parents and one grandparent, and a child with two parents and two grandparents.

## 6.3 Runtime and Heuristics

This section will focus on the performance of computing $d$-$k$-anonymity, with distance up to 5. First, we will study the effect of using the iterative algorithm and using heuristics on the runtime. We will compare runtimes when using different heuristics on ER and BA graphs with 1,000 vertices and average degree (or $m$ for BA graphs) {1, 2, 4, 8, 16, 32, 64, 128}. Then we will in more detail look at the runtime per iteration for graph models and the family network.

### 6.3.1 Heuristics

In Figure 6.8, the difference in runtime can be found when using the naive algorithm (as described in Section 4.2), the iterative algorithm (as described in Section 4.3), and the iterative algorithm with one of the two heuristics discussed in Section 4.4. Here we see that using the iterative approach and using one of the heuristics both result in a significant speed-up. Using a heuristic can result in a speed-up of a factor larger than 100 on ER graphs, and a factor larger than 1,000 on BA graphs. When comparing the iterative approach while using a heuristic with the naive approach, this even results in a speed-up larger than a factor 10,000 for most ER and BA graphs.




**(a)** Erdős Rényi **(b)** Barabási Albert

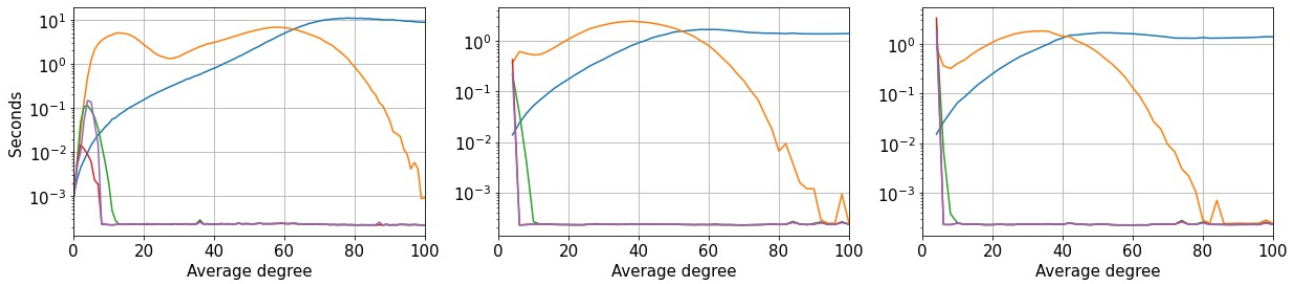
**Figure 6.8:** Runtime when using the naive or iterative algorithm, and using heuristics on ER and BA graphs with average degree and $m$ ranging from 1 to 128 and $|V| = 1{,}000$

Between the heuristics, no large difference in runtime is observed for these graphs. While the degree distribution is a more expensive heuristic, this is also a more powerful heuristic and able to reduce the set of feasible equivalence classes further in some cases. Apparently, this balances out in such a way, that it still results in similar runtimes on both ER and BA graphs.

### 6.3.2 Runtime: Graph Models

In Figure 6.9, the runtimes per iteration for ER, BA and HK graphs with 10,000 vertices can be found. Here it can be seen that in most cases the first and second iterations are the most expensive. During the first iteration, the equivalence check is the cheapest overall, since it looks at the 1-neighbourhood of vertices. However, since the equivalence partition consists of only one class at the start of this iteration, more vertices have to be compared during this iteration which results in longer runtimes. Especially after average degree 100 (ER), 80 (BA), 90 (HK), the 1-neighbourhoods of all vertices are unique, which causes that no work has to be done during the remaining iterations.

In the beginning, for graphs with low average degrees (0–10) for ER graphs, the other iterations are more expensive. Here the diameter of ER graphs is also larger, and more vertices are equivalent with respect to 1- and 2-$k$-anonymity. The second iteration is more expensive than the first one until average degree 62 (ER), 50 (BA) and 40 (HK). While the equivalence classes have been split into smaller classes

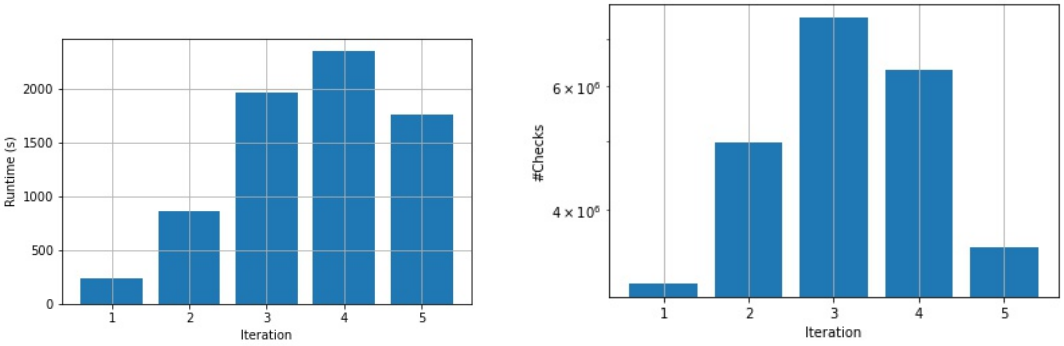



**(a)** Erdős Rényi **(b)** Barabási Albert **(c)** Powerlaw Cluster Graph

**Figure 6.9:** Runtime per iteration for ER (a) BA (b) and HK (c) graphs with 10,000 vertices and average degree ranging from 0 to 100

during the first iteration, the second iteration looks at a larger neighbourhood which makes it more expensive to check if vertices are equivalent. Apparently, this more expensive check results in a larger runtime for sparse graphs, for ER, BA, and HK. These results also show it is beneficial for the runtime to iteratively increase the neighbourhood, instead of starting at a distance $d$.

### 6.3.3 Runtime: Family Network




**(a)** Runtime per iteration **(b)** Number of equivalence checks

**Figure 6.10:** Runtime per iteration in the full family network

In Figure 6.10 the runtime per iteration (6.10a) and the number of equivalence checks, which is a comparison of two vertices, per iteration (6.10b) can be found. The first figure shows that the runtime per iteration first increases when the distance increases. This is as a result of the increased computation time of the equivalence checks; the $d$-neigbhourhood becomes larger when $d$ increases resulting in more expensive equivalence checks. However, at distance 5 the runtime is lower than in the previous 2 iterations. Figure 6.10b explains this result; at distance 5 the number of comparisons is much smaller than during the previous iterations. When fewer comparisons have to be made, this can also decrease the runtime. An explanation for the decrease in equivalence checks could be a combination of the smaller equivalence classes and that the heuristics can give a much smaller set of candidate classes at this distance. While the $d$-neighbourhoods at this distance are larger than for smaller distances, this results in a shorter runtime for this iteration, despite the increased expensiveness of the equivalence check. Similarly, for distance 1 we see very few isomorphism checks; this is likely caused due to the fact that at distance 1 not a lot of equivalence classes are created, therefore fewer comparisons are required overall to find the correct one for a given vertex.

# Chapter 7

# Conclusion

In this thesis, we have discussed various different measures for anonymity and positioned the new measure of $d$-$k$-anonymity in the context of existing literature. Since this measure is parametrized, one can easily set the desired strictness based on an estimation of how much information a possible attacker knows, and choose a right balance between anonymity and data utility. This makes $d$-$k$-anonymity a valuable addition to the already existing measures of $k$-anonymity.

We proposed a new algorithm to compute this measure and implemented it in C++. Combined with heuristics that resulted in a significant speed-up, we were able to measure the anonymity of well-known graph models with up to 10,000 vertices and the full family network of the Netherlands, which consists of 15.7 million vertices.

For all graph models, we saw that for 1-$k$-anonymity the anonymity decreases slowly as the density increases, and at some point all vertices become unique. When increasing the strictness to 2-$k$-anonymity, we saw that the anonymity decreases much faster which leads to one of the main conclusions of this thesis; in graph models, vertices become very quickly unique at distance 2.

In the family network, we saw that at distance 1 almost all vertices had a high anonymity. The anonymity decreases as the distance increases and at distance 5, a quarter of all vertices will require anonymization. However, in the second largest component, the anonymity was also much lower. This is likely as a result of the small graph size of 635 vertices.

We also found that the directionality of edges decreases the anonymity of vertices. This effect can greatly differ per vertex, but is not as strong as when the distance is increased from 1 to distance 2.

Furthermore, we looked at some of the most and least frequently occurring neighbourhoods in the family network. Here we found that common neighbourhoods often have a simple structure and vertices with low degrees, while rare neighbourhoods often have a high degree or a complex structure.

Since unique vertices occur in all graphs discussed, this family network would still have to be anonymized before it can be shared. Therefore, a next step could be to anonymize this network, or other real-world networks, to achieve $d$-$k$-anonymity. This could allow organizations, such as CBS, to share this data. However, this might be challenging due to the finding that for graph models most of the vertices are unique when measuring 2-$k$-anonymity.

Another possible avenue is to measure the anonymity of more networks or in different layers of the network of the Netherlands.

Furthermore, one can extend the implementation of our measure for anonymity to take into account labelled edges or labelled vertices. The addition of attributes in network data can possibly lead to even more interesting research.

# Bibliography


[1] A. Hundepool, J. Domingo-Ferrer, L. Franconi, S. Giessing, E. S. Nordholt, K. Spicer, and P.-P. De Wolf, *Statistical disclosure control*, vol. 2. Wiley New York, 2012.

[2] L. Willenborg and T. De Waal, *Elements of statistical disclosure control*, vol. 155. Springer Science & Business Media, 2001.

[3] J. van der Laan and E. de Jonge, "Producing official statistics from network data," in *Proceedings of the 6th International Conference on Complex Networks and their Applications, Book of Abstracts*, p. 288, 2017.

[4] L. Zou, L. Chen, and M. T. Özsu, "K-automorphism: A general framework for privacy preserving network publication," *Proceedings of the of the VLDB Endowment*, vol. 2, no. 1, pp. 946–957, 2009.

[5] M. van der Loo, "Topological anonymity in networks," *CBS internal report*, 2020.

[6] B. Zhou and J. Pei, "Preserving privacy in social networks against neighborhood attacks," in *2008 IEEE 24th International Conference on Data Engineering*, pp. 506–515, IEEE, 2008.

[7] P. Erdos, A. Rényi, *et al.*, "On the evolution of random graphs," *Publication of the Mathematical Institute of the Hungarian Academy of Sciences*, vol. 5, no. 1, pp. 17–60, 1960.

[8] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks," *Science*, vol. 286, no. 5439, pp. 509–512, 1999.

[9] P. Holme and B. J. Kim, "Growing scale-free networks with tunable clustering," *Physical Review E*, vol. 65, no. 2, p. 026107, 2002.

[10] S. Ji, P. Mittal, and R. Beyah, "Graph data anonymization, de-anonymization attacks, and de-anonymizability quantification: A survey," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 2, pp. 1305–1326, 2016.

[11] K. Liu and E. Terzi, "Towards identity anonymization on graphs," in *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, pp. 93–106, 2008.

[12] M. Hay, G. Miklau, D. Jensen, D. Towsley, and P. Weis, "Resisting structural re-identification in anonymized social networks," *Proceedings of the VLDB Endowment*, vol. 1, no. 1, pp. 102–114, 2008.

[13] W. Wu, Y. Xiao, W. Wang, Z. He, and Z. Wang, "K-symmetry model for identity anonymization in social networks," in *Proceedings of the 13th international conference on extending database technology*, pp. 111–122, 2010.

[14] J. Cheng, A. W.-c. Fu, and J. Liu, "K-isomorphism: privacy preserving network publication against structural attacks," in *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, pp. 459–470, 2010.

[15] L. Zhang and W. Zhang, "Edge anonymity in social network graphs," in *2009 International Conference on Computational Science and Engineering*, vol. 4, pp. 1–8, IEEE, 2009.

[16] R. Ford, T. M. Truta, and A. Campan, "P-sensitive k-anonymity for social networks.," *Proceedings of The 2009 International Conference on Data Mining*, vol. 9, pp. 403–409, 2009.

[17] M. Yuan, L. Chen, S. Y. Philip, and T. Yu, "Protecting sensitive labels in social network data anonymization," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 3, pp. 633–647, 2011.

[18] S. Das, Ö. Eğecioğlu, and A. El Abbadi, "Anonymizing weighted social network graphs," in *2010 IEEE 26th International Conference on Data Engineering (ICDE 2010)*, pp. 904–907, IEEE, 2010.

[19] X. Ying and X. Wu, "Randomizing social networks: a spectrum preserving approach," in *proceedings of the 2008 SIAM International Conference on Data Mining*, pp. 739–750, SIAM, 2008.

[20] P. Mittal, C. Papamanthou, and D. Song, "Preserving link privacy in social network based systems," *arXiv preprint arXiv:1208.6189*, 2012.

[21] A. Alavi, R. Gupta, and Z. Qian, "When the attacker knows a lot: The gaga graph anonymizer," in *International Conference on Information Security*, pp. 211–230, Springer, 2019.

[22] Y. Zhang, M. Humbert, B. Surma, P. Manoharan, J. Vreeken, and M. Backes, "Towards plausible graph anonymization," *arXiv preprint arXiv:1711.05441*, 2017.

[23] A. Sala, X. Zhao, C. Wilson, H. Zheng, and B. Y. Zhao, "Sharing graphs using differentially private graph models," in *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*, pp. 81–98, 2011.

[24] Q. Xiao, R. Chen, and K.-L. Tan, "Differentially private network data release via structural inference," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 911–920, 2014.

[25] D. Romanini, S. Lehmann, and M. Kivelä, "Privacy and uniqueness of neighborhoods in social networks," *arXiv preprint arXiv:2009.09973*, 2020.

[26] S. Horawalavithana, J. G. A. Flores, J. Skvoretz, and A. Iamnitchi, "Behind the mask: Understanding the structural forces that make social graphs vulnerable to deanonymization," *IEEE Transactions on Computational Social Systems*, vol. 6, no. 6, pp. 1343–1356, 2019.

[27] S. V. Vadamalai, *Lost in the Crowd: Are Large Social Graphs Inherently Indistinguishable?* University of South Florida, 2017.

[28] G. Csardi and T. Nepusz, "The igraph software package for complex network research," *InterJournal*, vol. Complex Systems, p. 1695, 2006.

[29] A. A. Hagberg, D. A. Schult, and P. J. Swart, "Exploring network structure, dynamics, and function using networkx," in *Proceedings of the 7th Python in Science Conference* (G. Varoquaux, T. Vaught, and J. Millman, eds.), pp. 11 – 15, 2008.

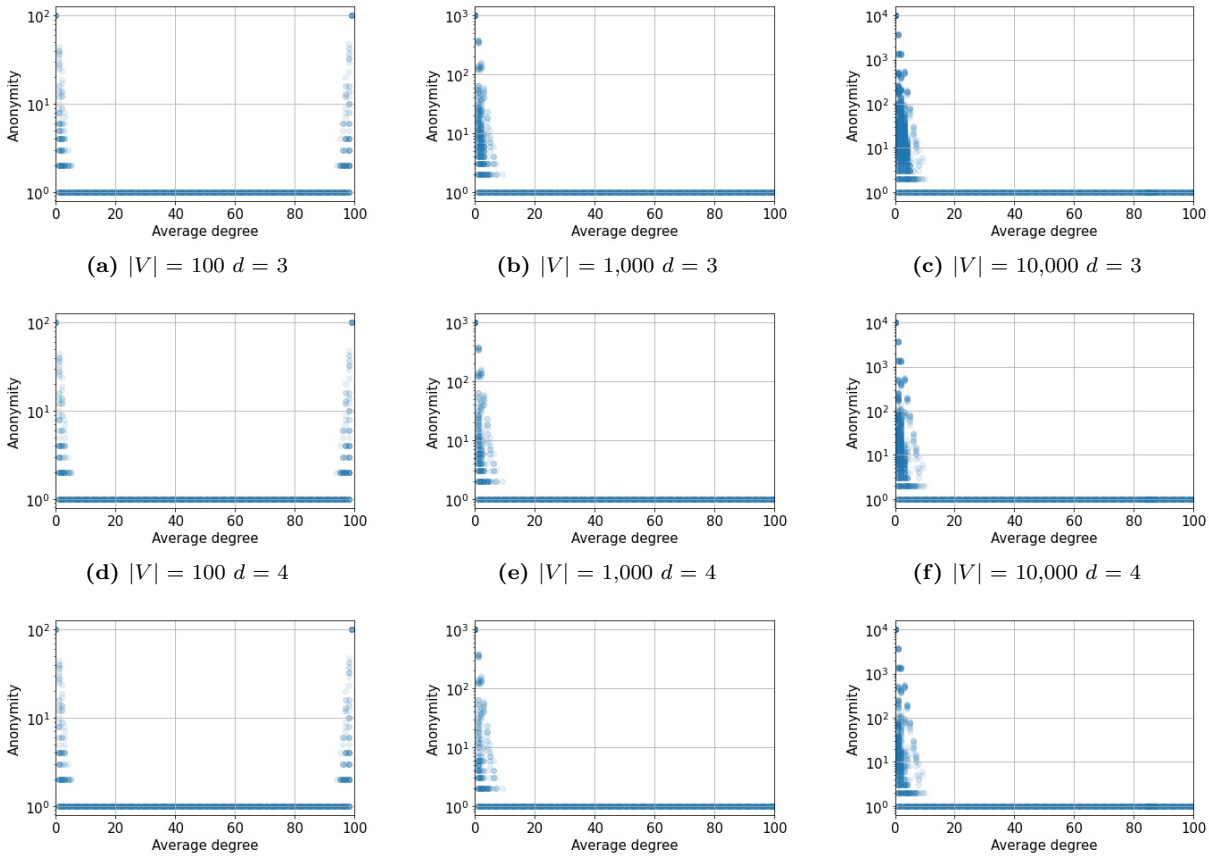
[30] B. D. McKay and A. Piperno, "Practical graph isomorphism, II," *Journal of Symbolic Computation*, vol. 60, pp. 94–112, 2014.

[31] M. Bastian, S. Heymann, and M. Jacomy, "Gephi: An open source software for exploring and manipulating networks," *International AAAI Conference on Weblogs and Social Media*, 2009.

# Appendices

# Appendix A

# Additional Results

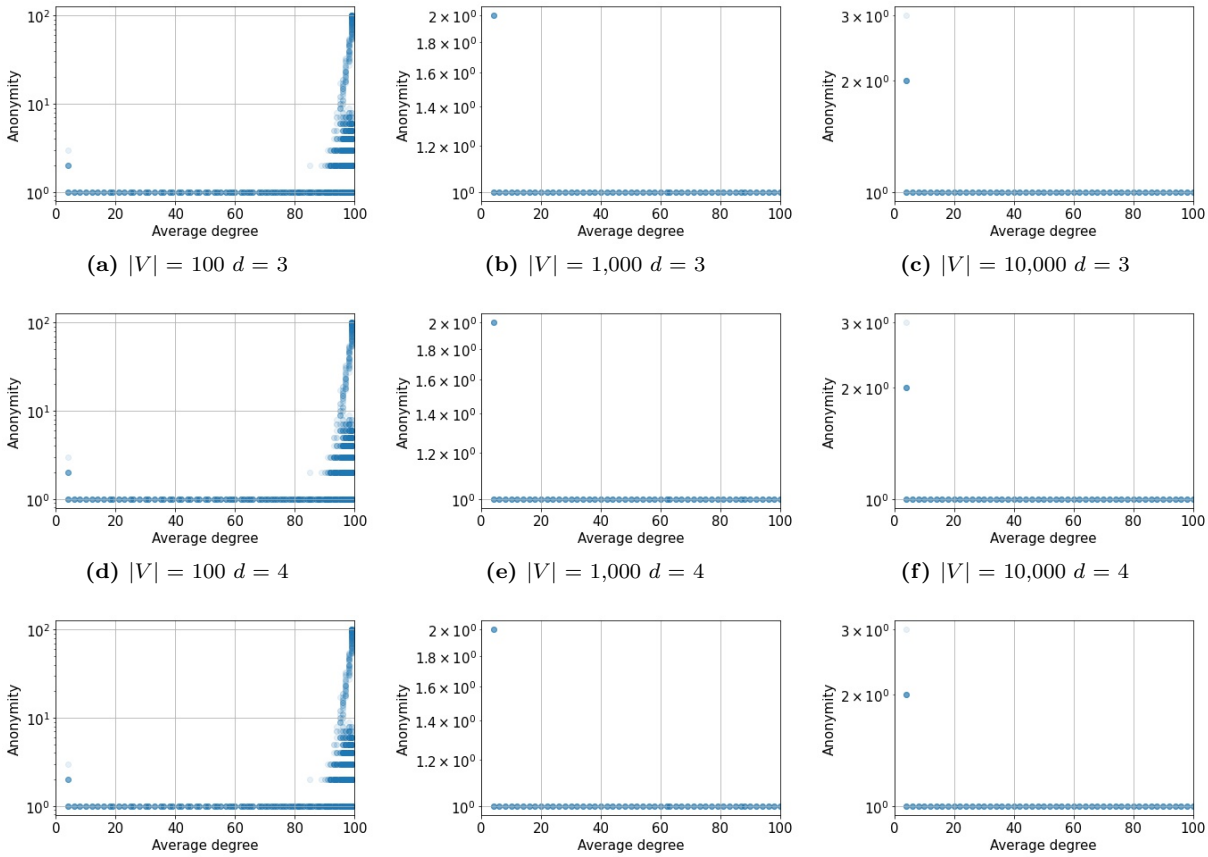



**(a)** $|V| = 100$ $d = 3$ **(b)** $|V| = 1{,}000$ $d = 3$ **(c)** $|V| = 10{,}000$ $d = 3$

**(d)** $|V| = 100$ $d = 4$ **(e)** $|V| = 1{,}000$ $d = 4$ **(f)** $|V| = 10{,}000$ $d = 4$

**(g)** $|V| = 100$ $d = 5$ **(h)** $|V| = 1{,}000$ $d = 5$ **(i)** $|V| = 10{,}000$ $d = 5$

**Figure A.1:** Anonymity distribution for ER graphs with respect to $d$-neighbourhoods, with $d \in \{3, 4, 5\}$.

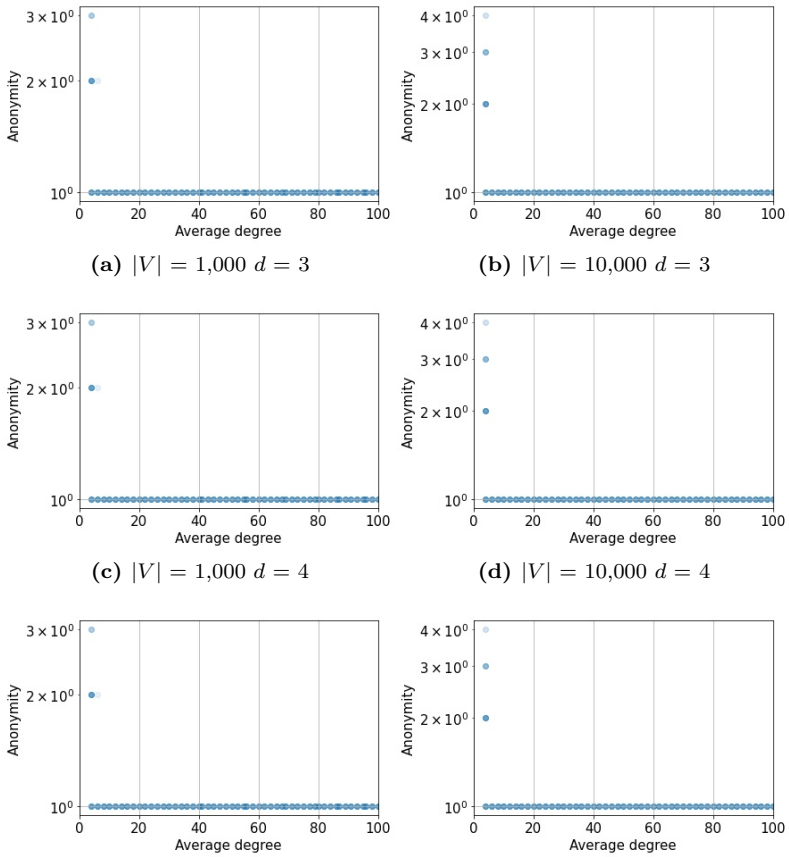



**(a)** $|V| = 100$ $d = 3$ **(b)** $|V| = 1{,}000$ $d = 3$ **(c)** $|V| = 10{,}000$ $d = 3$

**(d)** $|V| = 100$ $d = 4$ **(e)** $|V| = 1{,}000$ $d = 4$ **(f)** $|V| = 10{,}000$ $d = 4$

**(g)** $|V| = 100$ $d = 5$ **(h)** $|V| = 1{,}000$ $d = 5$ **(i)** $|V| = 10{,}000$ $d = 5$

**Figure A.2:** Anonymity distribution for BA graphs with respect to $d$-neighbourhoods, with $d \in \{3, 4, 5\}$.




**(a)** $|V| = 1{,}000$ $d = 3$ **(b)** $|V| = 10{,}000$ $d = 3$

**(c)** $|V| = 1{,}000$ $d = 4$ **(d)** $|V| = 10{,}000$ $d = 4$

**(e)** $|V| = 1{,}000$ $d = 5$ **(f)** $|V| = 10{,}000$ $d = 5$

**Figure A.3:** Anonymity distribution for HK graphs with respect to $d$-neighbourhoods, with $d \in \{3, 4, 5\}$.