Algorithms for Efficiently Computing Structural Anonymity in Complex Networks

RACHEL G. DE JONG and MARK P. J. VAN DER LOO, Leiden University, Statistics Netherlands, The Netherlands
FRANK W. TAKES, Leiden University, The Netherlands

This article proposes methods for efficiently computing the anonymity of entities in networks. We do so by partitioning nodes into equivalence classes where a node is k-anonymous if it is equivalent to $k-1$ other nodes. This assessment of anonymity is crucial when one wants to share data and must ensure the anonymity of entities represented is compliant with privacy laws. Additionally, in such an assessment, it is necessary to account for a realistic amount of information in the hands of a possible attacker that attempts to de-anonymize entities in the network. However, measures introduced in earlier work often assume a fixed amount of attacker knowledge. Therefore, in this work, we use a new parameterized measure for anonymity called d-k-anonymity. This measure can be used to model the scenario where an attacker has perfect knowledge of a node's surroundings up to a given distance d. This poses nontrivial computational challenges, as naive approaches would employ large numbers of possibly computationally expensive graph isomorphism checks. This article proposes novel algorithms that severely reduce this computational burden. In particular, we present an iterative approach, assisted by techniques for preprocessing nodes that are trivially automorphic and heuristics that exploit graph invariants. We evaluate our algorithms on three well-known graph models and a wide range of empirical network datasets. Results show that our approaches significantly speed up the computation by multiple orders of magnitude, which allows one to compute d-k-anonymity for a range of meaningful values of d on large empirical networks with tens of thousands of nodes and over a million edges.

CCS Concepts: • **Theory of computation → Graph algorithms analysis**; • **Design and analysis of algorithms**;

Additional Key Words and Phrases: Complex networks, graph algorithms, anonymity, privacy

ACM Reference format:
Rachel G. de Jong, Mark P. J. van der Loo, and Frank W. Takes. 2023. Algorithms for Efficiently Computing Structural Anonymity in Complex Networks. *ACM J. Exp. Algor.* 28, 1, Article 1.7 (August 2023), 22 pages.
https://doi.org/10.1145/3604908

This research was made possible by the Platform Digital Infrastructure SSH (http://www.pdi-ssh.nl).
Authors' addresses: R. G. de Jong, Leiden University, Statistics Netherlands (CBS), Niels Bohrweg 1, Leiden 2333 CA, The Netherlands; email: r.g.de.jong@liacs.leidenuniv.nl; M. P. J. van der Loo, Leiden University, Statistics Netherlands (CBS), Henri Faasdreef 312, Den Haag, The Netherlands; email: mpj.vanderloo@cbs.nl; F. W. Takes, Leiden University, Niels Bohrweg 1, Leiden 2333 CA, The Netherlands; email: f.w.takes@liacs.leidenuniv.nl.

1084-6654/2023/08-ART1.7 $15.00
https://doi.org/10.1145/3604908

1 INTRODUCTION

In order to obtain insights from networks in applied settings, it is desirable to have access to datasets that represent the real world as accurately as possible. Such empirical network datasets can be used to obtain insights into, for example, societally relevant issues including disease spread [1] and social segregation [4]. However, to comply with privacy laws, such data cannot be publicly shared unless the represented entities are properly anonymized.

In this work, we focus on networks where nodes represent entities of interest (such as people) and edges represent links between them. An attacker who attempts to deanonymize (a part of) the network may be able to use the network structure to uniquely identify entities [9, 20]. Therefore it is important to assess how revealing an entity's network structure is in possibly large real-world networks. In a subsequent step, the obtained information on each individual node's anonymity can be used to ensure the privacy of entities by means of anonymization. Through anonymization, the network structure is perturbed such that entities represented can no longer be identified based on the position in the network. However, this inevitably leads to an overall decrease in data utility: the extent to which the network can be reliably used for analysis.

Various network anonymity measures have been introduced in the literature [5, 9, 14, 20, 22, 30, 31, 33]. One category of methods is k-anonymity, which is also regularly used in the field of statistical disclosure control to characterize the anonymity of individuals in relational data [10, 18, 19, 23, 26, 28, 29]. In this category of approaches, each entity is said to be k-anonymous if there are at least $k-1$ other equivalent nodes in the network, which means these k nodes have similar structural positions. Consequently, when we assume a certain upper bound on the knowledge that an attacker has on the structural position of each entity, they will find at least k candidate nodes for each structural position.

Anonymity measures can be based on many different definitions for node equivalence. This results in measures with different levels of attacker knowledge. Some measures are based on local properties such as degree [14] or 1-neighborhood isomorphism [20, 31]. In addition, stricter measures taking into account degree distributions in neighborhoods [9] or automorphisms [30, 33] have been proposed. One shortcoming of these measures is that most measures assume a fixed level of attacker knowledge. Second, these measures might be either too lenient, not accounting for enough knowledge, and therefore not guaranteeing sufficient privacy, or too strict, accounting for a very large and therewith unrealistic amount of knowledge. When anonymizing data to achieve these strict levels of anonymity, a lot of data utility could potentially be lost due to subsequent pertubations to achieve anonymization. Hence, it is desirable to select a measure that accounts for the right amount of knowledge and therefore measures a suitable level of anonymity.

To account for the requirements above, we propose to use a new measure, d-k-anonymity, and corresponding algorithms. This measure, for which initial definitions and intuitions are sketched in [15], models a scenario where an attacker has perfect knowledge about the surroundings of a node up to, and including, distance d. The value of d is a parameter that can be set by the user which makes d-k-anonymity a tunable measure that can be used to measure anonymity in networks while accounting for different levels of attacker knowledge. With the parameter d, the measure generalizes previous work: when $d = 1$ it corresponds to 1-neighborhood isomorphism [20, 31], and when d is large enough, automorphism [30, 33]. At the same time, it builds upon other measures as it is more strict than degree [14], and adds to the measure introduced in [9] as it accounts for complete information about the d-neighborhood.

However, computing d-k-anonymity introduces computational challenges since naive approaches require a quadratic number of isomorphism checks to determine equivalence of the neighborhoods of all node pairs, which is a computationally difficult problem. In this work, we aim at overcoming these challenges by developing several approaches that speed up these computations by multiple orders of magnitude. The approaches consist of an iterative approach, the preprocessing of trivially automorphic nodes and heuristics that exploit graph invariants to avoid unnecessary computations. Overall, our approaches result in a speedup of multiple orders in magnitude compared to a naive approach and enable one to efficiently measure d-k-anonymity for a range of meaningful values of $d = 1$ up to $d = 3$ in large empirical networks with tens of thousands of nodes and over a million edges.

The remainder of this article is structured as follows. First, we discuss related work in Section 2. Then, in Section 3, we summarize the background information required for the remainder of the article. Thereafter, in Section 4, we formally define d-k-anonymity and discuss methods and heuristics for measuring it. In Section 5 we describe the data used in the experiments. Next, we discuss the experiments in Section 6 which focus on both the performance and analysis of the algorithms. Finally, Section 7 concludes this article and proposes directions for future work.

2 RELATED WORK

In this section, we briefly summarize the work related to our contribution, focusing on the different measures for anonymity. For a generic overview on anonymity in networks, we refer the reader to [11].

There are at least two large streams of research on anonymity in networks — differential privacy and k-anonymity — both originally designed for assessing anonymity in relational data [10, 18, 19, 23, 26, 28, 29]. The first, differential privacy, gives possibly randomly permuted answers to user queries about the network to satisfy certain disclosure constraints. As a result, the anonymity of entities represented is preserved. The second, which is the focus of this article, uses the concept of k-anonymity. In standard SDC methodology, k-anonymity [18, 19, 23, 26] assigns a risk of disclosure based on occurrences of attribute values across records. Extensions of k-anonymity, such as (α, k)-anonymity [29] extend the general notion of k-anonymity by enforcing diversity in the equivalence classes. For networks, if each node in the network is equivalent to at least $k-1$ other nodes, the network is called k-anonymous. An important difference between the two approaches is that in differential privacy, the network is perturbed to satisfy certain disclosure constraints. In k-anonymity, measuring the anonymity of nodes is separated from the process of anonymization.

In order to measure k-anonymity, we have to define what it means for two nodes to be equivalent. The simplest measure is based on the degree of a node [14], in which case nodes are equivalent if they have the same number of connections. Stricter measures that take into account more structural information are based on 1-neighborhood isomorphism [20, 31], which accounts for the direct neighborhood of a node, or automorphism [30, 33], which accounts for the entire connected component that the considered node is part of. These measures assume a fixed level of attacker knowledge and whereas 1-neighborhood isomorphism is relatively lenient, automorphism is very strict. A measure that can model different levels of attacker knowledge is introduced by Hay et al. in [9]. This measure iteratively compares degree distributions in increasingly larger neighborhoods of a node. However, by only counting degrees, the exact underlying structure, which is accounted for in 1-neighborhood isomorphism and automorphism, is lost.

The measure of d-k-anonymity (not to be confused with dK-graphs as introduced in [16]) used in this work improves on the work of [9] and allows one to model different levels of knowledge by combining automorphism with a tunable measure that enables the user to account for perfect knowledge of a node's surroundings up to distance d. Moreover, this measure accounts for more information than degree [14] and allows the user to differentiate in strictness, ranging from the level of 1-neighborhood isomorphism as defined in [20, 31] to automorphism as proposed in, e.g., [30, 33]. This differentiation can be selected in d-k-anonymity by the user by means of a parameter

d. This article thus unifies a number of existing lines of research, proposing a tunable measure that can be used to account for different levels of attacker knowledge.

3 PRELIMINARIES

In this section, we introduce definitions and notation required to understand the remainder of the article. First, we focus on graph terminology. Second, we discuss k-anonymity and different definitions for equivalence.

3.1 Graph Terminology

We define a *graph* or *network* $G = (V, E)$ as a set of *nodes* V and *edges* E where each edge is an unordered pair of nodes $\{v, w\}$ with $v, w \in V$. We let $|V|$ denote the number of nodes and $|E| \leq \binom{|V|}{2}$ the number of edges. We define the degree of a node as $degree(v) = |\{w : \{v, w\} \in E\}|$, the average degree as $\frac{2|E|}{|V|}$ and the network density as $\frac{|E|}{|V|(|V|-1)/2}$. An example graph can be found in Figure 1.

The distance between two given nodes $v, w \in V$ is denoted $distance(v, w)$. This equals the minimum number of edges that needs to be traversed to get from node v to node w. Since the graph is undirected, it holds for every pair of nodes that $distance(v, w) = distance(w, v)$. It follows that $distance(v, v) = 0$, and if there is no path between two nodes we define $distance(v, w) = \infty$. The latter is the case when nodes are in different *components*. Within a component, it holds that for each pair of nodes, $distance(v, w) < \infty$. Empirical networks often have one large component, the *giant component*, and many small components. The largest distance between any two nodes in the graph that does not equal ∞ is defined as the *diameter* $D(G)$; this equals the length of the longest shortest path between any two given nodes.

When we look at the surroundings of a node v containing all nodes w such that $distance(v, w) \leq d$, we look at the d-neighborhood of the node, as defined below. When $d = 1$ this is also referred to as the *ego network*.

Definition 3.1 (d-Neighborhood). Given a graph $G = (V, E)$ and a node $v \in V$, we define the d-neighborhood of v, $N_d(v)$, as the graph $G' = (V', E')$ where:

— $V' = \{w \in V \ : \ distance(v, w) \leq d\}$.
— $E' = \{\{u, w\} \in E \ : \ u, w \in V'\}$.

In order to determine whether two d-neighborhoods are equivalent, we can use the notion of *isomorphism*. We say that two graphs or neighborhoods are isomorphic if their structures are indistinguishable. Note that to illustrate how these measures can be used to determine equivalence, Figure 1 includes an example figure that further explains each of the measures discussed.

Definition 3.2 (Graph Isomorphism). Given two graphs $G = (V, E)$ and $G' = (V', E')$, a graph isomorphism is defined as a bijective function $\phi : V \to V'$ such that for each $v, w \in V$ it holds that $\{\phi(v), \phi(w)\} \in E'$ iff $\{v, w\} \in E$.

A special case of isomorphism is an *automorphism*, denoted by γ, analogously to ϕ. An automorphism is an isomorphism from a graph onto itself. We let G^γ denote the transformation of G under γ. For every graph, there is always at least one automorphism; the function that maps all nodes onto themselves. When there is an automorphism mapping two different nodes onto each other, the nodes are said to be in the same *orbit*. As is the case with isomorphic functions, these nodes are indistinguishable based on their precise structural position in the graph.

Determining if two graphs are isomorphic is a computationally difficult problem. One method to determine whether two graphs are isomorphic is by comparing their *canonical labelings* [17].

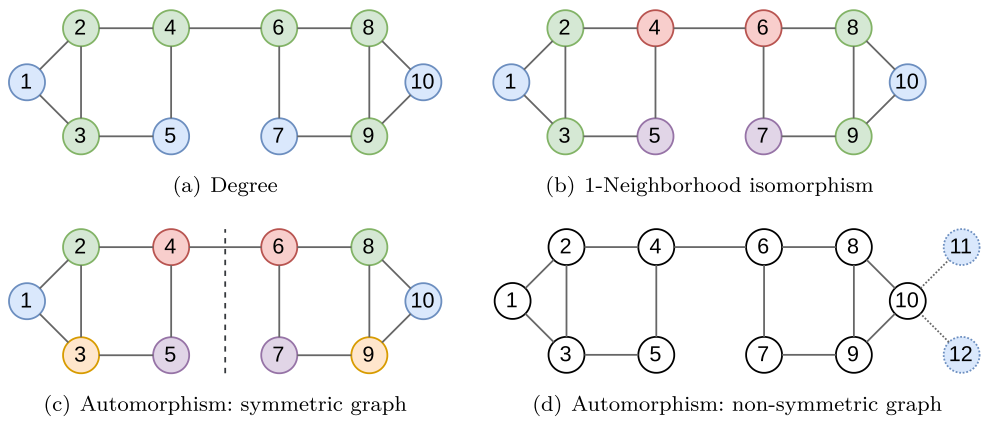

(a) Degree
(b) 1-Neighborhood isomorphism
(c) Automorphism: symmetric graph
(d) Automorphism: non-symmetric graph

Fig. 1. Graphs with partitions based on different measures for equivalence: degree (a), 1-neighborhood-isomorphism (b) and automorphism (c, d). Nodes in the same equivalence class have the same color, white nodes are unique.

These labelings are generated in such a way that two canonical labelings are equal if and only if the graphs are isomorphic. This canonical labeling is hence one of the possible graphs that is isomorphic to the original graph. For details about the employed approach to canonical labeling and for further information on how it is computed, we refer the reader to [17].

Definition 3.3. The canonical labeling of a graph G is the image of a function C such that given graphs $G = (V, E)$ and $G' = (V', E')$, and any automorphic function γ the following two properties hold:

— $C(G^\gamma) = C(G)$.
— $C(G) = C(G')$ iff G is isomorphic to G'.

3.2 Anonymity Measures

When measuring k-anonymity in a graph, the first conceptual step is to partition the set of nodes V into *equivalence classes*. A partition P consists of disjoint subsets of V such that their union is precisely V. Each node v occurs in exactly one equivalence class referred to by P^v, and for two nodes $v, w \in P^v$ we say that they are equivalent. In this case, it follows that $P^v = P^w$. We define k-anonymity for nodes and graphs using the definition below.

Definition 3.4 (k-Anonymity). Given a graph $G = (V, E)$ and equivalence partition P:

— A node $v \in V$ is k-anonymous if $|P^v| = k$.
— The graph G is k-anonymous if for all nodes $v \in V$ it holds that $|P^v| \geq k$.

In the previous section, various graph properties have been discussed that are proposed in previous works as a measure for anonymity. In Figure 1, the nodes are partitioned based on three of these; in each figure equivalent nodes have the same color, whereas white nodes are 1-anonymous or unique and occur in an equivalence class of size 1. The first and simplest measure is based on the degree of nodes [14], as shown in Figure 1(a).

Second, Figure 1(b) shows a slightly stricter measure being 1-neighborhood isomorphism. Here nodes are equivalent if their 1-neighborhoods are isomorphic [20, 31]. The measure therefore results in a higher number of equivalence classes of smaller sizes. This takes account of the degree of nodes and other structural properties such as triangles, which occur when two direct neighbors are connected, e.g., nodes 1, 2, and 3 in the figure. These properties cannot be accounted for when only taking degrees into account in the assessment of equivalence.

Third, we consider a measure that accounts for the structure of the entire component a node occurs in, being k-automorphism [30, 33]. Here nodes are equivalent if they are in the same orbit. This is a very strict measure; for a graph to be k-anonymous, all its components have to be symmetric in at least $k-1$ points. This is the case in Figure 1(c) where each node, and therewith the graph, is 2-anonymous. In this figure, nodes with the same color are in the same orbit. However, in empirical networks, this is very unlikely; even one node or edge can undo this symmetry resulting in a large increase in the number of unique nodes, as shown in Figure 1(d). In this figure only nodes 11 and 12 are still equivalent. Henceforth, we will call such nodes *twin nodes*: they are connected to the same nodes, without any other connections and are, as a result, structurally identical. We will further elaborate on this notion in Section 4.4.

Definition 3.5 (Twin Node). Given a graph $G = (V, E)$, two nodes $v, w \in V$ $(v \neq w)$ are twins if $N_1(v) \setminus \{v\} = N_1(w) \setminus \{w\}$ and $\{v, w\} \notin E$.

4 APPROACH

In this section, we define the measure of d-k-anonymity and introduce both a naive and iterative approach to compute it. Then, in order to further reduce computation time, we introduce a preprocessing step based on twin nodes, and two heuristics based on graph invariants.

4.1 d-k-Anonymity

We define d-k-anonymity by saying that two nodes are equivalent if they have the same structural position in their respective (isomorphic) d-neighborhoods [15]. This implies that when an attacker has perfect information about the neighborhood of a node up to and including distance d in a d-k-anonymous graph, there will be at least k possible candidates to which that node is equivalent. Here d is an input parameter for the measure, and k is a constraint for being anonymous for which the value can be derived from the resulting equivalence classes.

Definition 4.1 (d-k-Anonymity). Given a graph $G = (V, E)$, nodes $v, w \in V$ are d-equivalent if:

— At least one isomorphism ϕ between $N_d(v)$ and $N_d(w)$ exists.
— For one such isomorphism it holds that $\phi(v) = w$.

A node is d-k-anonymous if it is d-equivalent to $k-1$ other nodes, and a graph is d-k-anonymous if all nodes are at least d-k-anonymous.

The parameter d can be used to model the amount of information available to an attacker. When $d = 0$, no information is included and all nodes are equivalent. When $d = 1$ it coincides with *1-neighborhood isomorphism* [20, 31] and when $d \geq D(G)$, it coincides with k-*automorphism* [30, 33] (both discussed in Section 3.2). The latter follows from that the d-neighborhood consists of the entire component that the considered node belongs to; when the graph consists of one component this equals the entire graph. In all other cases, d-k-anonymity can be seen as an approximation of automorphism; equivalent nodes are indistinguishable if the full structural position of the nodes up to distance d is known.

4.2 Naive Algorithm

To measure d-k-anonymity in a graph, we determine for each pair of nodes if they are d-equivalent using Definition 4.1. This can be done with the notions of canonical labelings and orbits as defined in Section 3.1. When the d-neighborhoods of the nodes have the same canonical labeling, they are isomorphic. In order for the second requirement to hold, there should be at least one isomorphism that maps the considered nodes onto each other. This holds when the isomorphism maps the currently considered node in the first graph onto a node in the second graph that is in the same

ALGORITHM 1: NAIVE-D-K-ANONYMITY

```
1  Input: Graph $G = (V, E)$, equivalence class $eq = \{V\}$, distance $d$
2  $P = \emptyset$, $cache = \{\}$
3  $eq, twin = remove\_twins(eq)$                        ▹ Optional (Section 4.4)
4  for $v_1$ in $eq$ do                                  ▹ For each node, find the equivalence class
5      $P_{selection} = preprocess(P,\ v_1)$             ▹ Optional (Section 4.5)
6      $make\_new\_partition$ = True
7      if $P_{selection} \neq \emptyset$ then
8          $can_1, v_{1pos} = canonical\_labeling(N_d(v_1))$   ▹ Compute first canonical labeling [17]
9          $hcan_1 = hash(can_1)$                        ▹ Determine hash of labeling
10         $cache[v_1] = (hcan_1, v_{1pos})$
11     end
12     for $eq'$ in $P_{selection}$ do
13         $v_2 = eq'[0]$                                ▹ Get first added node from current class
14         if $v_2 \in cache$ then
15             $hcan_2, v_{2pos} = cache[v_2]$           ▹ Get from cache
16         else
17             $can_2, v_{2pos} = canonical\_labeling(N_d(v_2))$   ▹ Otherwise compute labeling
18             $hcan_2 = hash(can_2)$
19             $cache[v_2] = (hcan_2, v_{2pos})$
20         end
21         if $hcan_1 == hcan_2$ and $same\_orbit(v_1, v_{2pos})$ then
22             $eq' = eq' \cup \{v_1\}$                  ▹ Add equivalent node to class
23             $make\_new\_partition$ = False            ▹ Finished for current node
24             break
25         end
26     end
27     if $make\_new\_partition == True$ then
28         $P = P \cup \{\{v_1\}\}$                      ▹ If not equivalent, create new class
29     end
30 end
31 $P_{new} = add\_twins(P_{new}, twin)$                 ▹ Optional (Section 4.4)
32 return $P$                                            ▹ Return equivalence partition
```

orbit as the node it is compared to. When these two requirements hold, the nodes are d-equivalent. This allows the full set of nodes to be partitioned into equivalence classes, whose size k can subsequently be used to determine whether the nodes and the graph are d-k-anonymous, cf. Definition 4.1.

The pseudocode in Algorithm 1 presents the high-level algorithm to compute the aforementioned equivalence classes of nodes in the graph. The main part of this algorithm consists of two nested loops. The outer loop, on lines 4–30, iterates over each node v_1 in the graph, while the inner loop, on lines 12–26, checks whether v_1 belongs to one of the equivalence classes created thus far. If v_1 does belong to a previously created equivalence class, line 22 adds v_1 to that class. Otherwise, line 28 creates a new equivalence class consisting of v_1. Algorithm 1 additionally contains three steps to speed up the computation. Lines 3 and 31 perform the pre- and postprocessing of twin nodes to reduce the number of nodes in eq during computation. This step will be described in Section 4.4. Second, the preprocessing step in line 5 filters out certain candidate equivalence classes to reduce the number of comparisons. In Section 4.5 we discuss several variants of this preprocessing step. The hashing step in lines 9 and 18 is needed for comparison: two neighborhoods are isomorphic only if the hash values of their canonical labelings are equal. To obtain this hash, the HASHGRAPH_SG function from [17] is used. In lines 10 and 19 hashes are stored in a cache that is used in lines 14 and 15 to avoid duplicate computations of canonical labelings.

If a node is equivalent to one node in the equivalence class, it is equivalent to all nodes in it; therefore we have to compare each node to only one node per equivalence class selected in line 13. Equivalence of two nodes is assessed in line 21. Two requirements are checked: (1) whether the hash of the canonical labelings (precomputed using the function SPARSENAUTY from [17] in lines 8 and 17) are equal and (2) if the nodes considered (v_1, v_2) are in the same orbit. The function *same_orbit* first maps v_2 onto a node in the canonical labeling (v_{2pos}) and then onto a node v_2' in $N_d(v_1)$. Next, it checks if v_1 and v_2' occur in the same orbit.

If the nodes are equivalent, we add v_1 to this class in line 22. Otherwise, we create a new equivalence class that consists solely of this node in line 28, which is also the default action if the set of equivalence class candidates $P_{selection}$ is empty. The result is a partitioning of the nodes into equivalence classes, which, as described above, can subsequently be used to assess d-k anonymity for all nodes.

The cache used to avoid redundant computations of canonical labelings consists of two parts. We store for each equivalence class (1) a hash integer generated by the Nauty function HASHGRAPH_SG which returns a unique integer representing the given canonical labeling, and (2) the position of the node in this canonical labeling (necessary for the second condition of Definition 4.1). The first is used to determine if the neighborhoods are isomorphic and the second to determine if the nodes are in the same orbit. Therefore, when both are known, comparing two nodes will require only two comparisons and access to a few array elements (for the orbit) to determine if two nodes are d-equivalent. As a result, the canonical labeling of each node has to be computed at most once, and any further comparisons can be done in constant time. The amount of memory used for storing these values is only linear in the number of nodes. Note that a version without cache would omit lines 10, 14, 15, and 19 in Algorithm 1.

4.3 Iterative Algorithm

Algorithm 1 has one disadvantage. Since the d-neighborhoods of nodes can be relatively large, the computation of canonical labelings in lines 8 and 17 can be relatively expensive. At the same time, all nodes must be compared at least once, which requires the canonical labeling of the d-neighborhood for each node to be computed. To counteract this disadvantage, we introduce the iterative approach illustrated in Algorithm 2.

This approach uses the property that when nodes are $(d + 1)$-equivalent, they need to be d-equivalent [15]. Hence, to determine whether two nodes are $(d + 1)$-equivalent, we only compare them when they have already been found to be d-equivalent. Using this approach, we essentially let the neighborhood radius increase iteratively. In the first iteration, the algorithm requires many comparisons, but these are relatively fast. During later iterations, the size of neighborhoods compared increases, which yields a strong growth in the computational time necessary to compute canonical labelings. However, as a result of previous iterations, smaller equivalence classes are obtained and fewer comparisons are required overall. The effect of this approach on the runtime will be investigated in Section 6.

4.4 Twin Nodes

As briefly discussed in Section 4.2, the algorithm includes an optional preprocessing step that detects twin nodes. Twin nodes are nodes with the exact same direct neighbors (Definition 3.5 in Section 3.2). In empirical networks, they can be observed frequently as a result of preferential

ALGORITHM 2: ITERATIVE-D-K-ANONYMITY

```
1  Input: Graph $G = (V, E)$, equivalence class $eq = \{V\}$, distance $d$
2  $eq, twin = remove\_twins(eq)$                          ▷ Optional (Section 4.4)
3  $P_{old} = eq$
4  for $cur\_dist$ = 1 to $d$ do                             ▷ Increase distance
5      $P_{new} = \emptyset$
6      for $eq\_it$ in $P_{old}$ do                          ▷ Split each equivalence class
7          $P_{new} = P_{new} \cup$ NAIVE-D-K-ANONYMITY($G$, $eq\_it$, $cur\_dist$)   ▷ Algorithm 1
8      end
9      $P_{old} = P_{new}$
10 end
11 $P_{new} = add\_twins(P_{new}, twin)$                    ▷ Optional (Section 4.4)
12 Return: $P_{new}$                                        ▷ Return equivalence partition
```

attachment and the skewed degree distribution [2] which results in a large number of nodes with a low degree. An example of twin nodes are nodes 11 and 12 in Figure 1(d).

THEOREM 4.2. *Given a graph $G = (V, E)$ and nodes $v, w \in V$ $(v \neq w)$. If nodes v, w are twins, then they are in the same orbit.*

Twin nodes have a special property: because they are connected to the same nodes, they are in the same orbit (for a formal proof, see Appendix A). Therefore, we know that twin nodes are equivalent with respect to d-k-anonymity for any given distance d. When two nodes are twins, and the equivalence class of one of the nodes has been identified, then the other node can be added to the equivalence class of its twin.

We preprocess these twin nodes in Algorithm 2, line 2 by first partitioning the nodes with the same direct neighbors into the same set. To find the twin nodes more efficiently, we keep track of a map containing a set of neighbors and the corresponding node. If for a given node the set of neighbors already appears in the map, we have found a twin for the current node, and add it to a set of nodes that are twins of each other. Otherwise, if the set of neighbors is not included in the map, we add a new entry for the current node. This approach results in a time complexity of $O(|V|log|V|)$ to find all twin nodes. From each set of nodes that are twins, one node is used in the computation of the equivalence classes and the remainder of the twin nodes are not taken into account during computation. To balance the gain and computational time of this step, it is possible to limit the computation to a fixed number of neighbors (see Section 6.1 and 6.3.2). After execution of the algorithm, each skipped twin node is added to the correct equivalence class containing the node that contains its twin in Algorithm 2, line 11.

4.5 Heuristics

The last method to speed up the computation is by the use of heuristics. Instead of comparing each node to all equivalence classes, we use a heuristic to make a selection of equivalence classes to compare it to (Algorithm 1, line 5). This results in a smaller set of candidate equivalence classes. Furthermore, when there are no candidate classes for a node, no canonical labeling computation is required. Only when this new equivalence class is a candidate for a different node, the labeling will be computed and stored in the cache in lines 17–19.

The heuristics used are cases of a graph invariant: properties that two graphs have in common if they are isomorphic. Since isomorphism is the first requirement for d-k-anonymity, we have chosen to, before comparing canonical labelings, filter the set of candidate classes using the following heuristics:

— COUNT: the number of nodes and edges in the d-neighborhood.
— DEGREES: the degree distribution in the d-neighborhood.

Values for the graph invariants are computed during each iteration by first obtaining the d-neighborhood, and then computing the values. The first invariant is the easiest to compute and compare, yet, due to the iterative application in Algorithm 2, very powerful as it additionally captures other properties such as the distance distribution of nodes and edges in the d-neighborhood. It should be noted that the computation time of determining these heuristics is constant or linear in the number of nodes given that neighborhoods are stored in an adjacency list format. This is negligible compared to that of the more expensive canonical labeling computations.

The degree distribution is more time-consuming to compute, but also more informative. Note that from the degree distribution, the number of nodes and edges can be derived, and therefore the same information is included; the number of nodes equals the number of degrees that is counted, and the number of edges equals half of the sum of each degree multiplied by its occurrence. We empirically analyze the effect of both heuristics on performance in Section 6.

5 DATA

For the experiments, we applied the algorithms introduced in Section 4 to artificial networks generated using three different commonly used graph models, as well as 32 empirical networks gathered from various open source repositories.

The graph models are used to assess how the anonymity and runtime change as the density increases. We have chosen three common graph models that reflect different aspects of empirical networks. The first and simplest graph model is the **Erdős-Rényi** (**ER**) model [6] which adds each possible edge with a probability $0 < p < 1$. Second, the **Barabási-Albert** (**BA**) model [3] accounts for preferential attachment. In constructing this type of graph, each added node is connected to m different nodes where nodes are more likely to attach to nodes with a high degree. This results in a skewed degree distribution that is often observed in empirical networks [2]. Third, the **Wats-Strogatz** (**WS**) model [27] accounts for small average path lengths and clustering. These graphs are generated by first placing the nodes in a circle and then connecting each node to its $m \geq 2$ nearest neighbors. Then each edge is rewired with probability $p_r = 0.5$, which is a common value that is for example also used in [20]. This balances the values $p_r = 0.0$, which results in a circle graph, and $p_r = 1.0$ which resembles an ER graph.

The empirical networks and their topological characteristics can be found in Table 1. This table contains for each network the name (leftmost column), the type (rightmost column), structural characteristics (columns two to six), the fraction of unique nodes using 1-k-anonymity (column seven) and the fraction of twin nodes skipped by configurations that include the twin node step (column eight). Networks used are of various categories and have a variety of sizes, densities and other properties. The networks can consist of multiple components. While in some networks, such as biological networks, anonymity perhaps does not need to be measured for privacy reasons, we have chosen to include these since it could lead to insights into how different types of network structures show different distributions of anonymity. For all networks, directionality and any additional metadata are ignored. All networks are openly available and can be found in the corresponding repository cited in the leftmost column.

6 EXPERIMENTS

In this section, we test and compare the performance of the various algorithms and configurations to compute d-k-anonymity. First, we summarize the experimental setup used. Second, we discuss the results on graph models including the performance of different configurations and the

Table 1. Descriptives of the Used Empirical Networks

Network	$\|V\|$	$\|E\|$	Average degree	Average distance	D(G)	Fract. unique $d = 1$	Fract. twins skipped	Type
Radoslaw emails [12]	167	3,251	38.90	1.9	5	0.77	0.036	Communication
Primary school [25]	236	5,899	50.0	1.8	3	1.00	0.000	Human contact
Moreno innov. [12]	241	923	9.1	2.5	5	0.63	0.004	Communication
Gene fusion [12]	291	279	1.9	3.8	9	0.02	0.505	Bio
Copnet calls [24]	536	621	8.4	3.0	6	0.04	0.018	Communication
Copnet sms [24]	568	697	2.4	7.3	20	0.04	0.067	Communication
Copnet FB [24]	800	6,429	16.1	6.3	7	0.81	0.001	Facebook
FB Reed98 [21]	962	18,812	39.1	2.4	6	0.91	0.006	Facebook
Arenas email [12]	1,133	5,451	9.6	3.6	8	0.49	0.022	Communication
Network science [12]	1,461	2,742	3.8	6.3	17	0.07	0.024	Co-autorship
FB Simmons81 [21]	1,518	32,989	43.5	2.6	7	0.91	0.002	Facebook
DNC emails [12]	1,893	4,466	4.7	3.3	8	0.11	0.628	Facebook
Moreno health [12]	2,539	10,455	8.2	4.5	10	0.33	0.001	Human social
FB Wellesley22 [21]	2,970	94,900	63.9	2.5	8	0.93	0.000	Facebook
Bitcoin alpha [21]	3,783	14,124	7.5	3.6	10	0.20	0.239	Online social (trust)
GRQC collab. [13]	5,242	14,496	5.5	6.1	17	0.13	0.058	Co-autorship
FB Carnegie49 [21]	6,637	249,967	75.3	2.7	8	0.90	0.003	Facebook
Pajek Erdős [12]	6,927	11,850	3.4	3.7	4	0.07	0.658	Co-autorship
DT interaction [32]	7,341	15,138	4.1	5.9	18	0.00	0.432	Bio
DG assoc. [32]	7,813	21,357	5.5	4.2	8	0.01	0.469	Bio
FB GWU54 [21]	12,193	469,528	77.0	2.8	9	0.90	0.002	Facebook
Anybeat [21]	12,645	49,132	7.8	3.1	10	0.15	0.443	Online social
CE-CX [21]	15,229	245,952	32.3	3.7	13	0.57	0.005	Bio
Astro Physics [21]	18,771	198,050	21.1	4.3	14	0.37	0.015	Co-autorship
FB BU10 [21]	19,700	637,528	64.7	3.0	9	0.89	0.002	Facebook
FB Uillinois [21]	30,664	1,048,574	68.4	3.1	9	0.90	0.001	Facebook
Enron email [13]	36,692	183,831	10.0	4.0	13	0.19	0.264	Communication
FB Penn [21]	41,536	1,362,220	65.6	3.1	8	0.88	0.001	Facebook
FB wall 2009 [12]	46,952	193,494	8.2	5.7	18	0.19	0.067	Communication
Brightkite [21]	58,228	214,078	7.4	4.9	18	0.16	0.157	Online social
The marker cafe [7]	69,413	1,644,849	47.4	3.0	9	0.42	0.167	Human contact
Slashdot zoo [12]	79,116	467,731	11.8	3.9	12	0.12	0.204	Online social

The average distance is obtained from metadata in the repository. When not available, the average distance is approximated by computing the distance from 200 random nodes to 200 random nodes. For each network, we list the fraction of unique nodes at distance 1 and the fraction of skipped nodes that have at least one twin.

anonymity distribution. Third, we focus on the empirical networks and discuss the performance of heuristics, anonymity distributions and their effect on the computation time. This presents the reader with a complete overview of the overall performance of the proposed approaches on graph models and empirical networks.

6.1 Experimental Setup

For the experiments, we implemented the d-k-anonymity measure described in Section 4 in C++ and used the Nauty framework [17] to compute the canonical labelings and orbits. Our source code is available at https://github.com/RacheldeJong/dkAnonymity.

We executed experiments on both the three graph models with 1,000 nodes each and the empirical networks as discussed in Section 5. For all experiments, we computed d-k-anonymity for $d = 5$. Compared to the observed average path lengths and diameters, this is a large value that accounts for a lot of attacker knowledge, likely much more than available in a realistic scenario. In the comparative performance experiments in Section 6.3, apart from $d = 5$, we also report on results for $d = 2$, for which the best configurations are able to finish in the allotted amount of time for all networks considered. More importantly, this value constitutes a meaningful parameter setting, going well beyond existing 1-neighborhood isomorphism approaches (e.g., as proposed in [20, 31]) in terms of strictness, while staying away from neighborhoods consisting of thousands of nodes, a less realistic attacker scenario often encountered when choosing $d = 5$ in empirical networks. For more discussion on how the various parameter settings essentially mimic previous approaches to computing anonymity, see Section 4.1.

We used the configurations corresponding to the approaches discussed in Section 4. For the graph models, the following configurations were used:

— NAIVE: immediately compute for $d = 5$ (Algorithm 1, no heuristics).
— ITERATIVE: iteratively increase neighborhood radius (Algorithm 2, no heuristics).
— COUNT: ITERATIVE + the number of nodes and edges heuristic.
— DEGREES: ITERATIVE + the degree distribution heuristic.

To account for the nondeterminism in the generative process of the graph models, the average over 10 graph realizations as generated by NetworkX [8] is reported. For each graph, we have used a time limit of one hour. The twin node preprocessing step, and thus the TWIN configuration are not included in the initial results because of the linearly increasing expected minimum degree in the discussed graph models.

For the empirical networks, we have used the five configurations below. For the last three configurations, we additionally use the twin node preprocessing step which finds all twin nodes with a degree of at most 5. This cut off is chosen based on the low fraction of twin nodes found with a degree larger than 5 (see Section 6.3.2 and Figure 6(b)) and the substantially longer computation time required to detect these. For the empirical networks, we used a time limit of three hours and report the average runtime and standard deviation over 5 runs. Subsequent results reported include values obtained within this time limit.

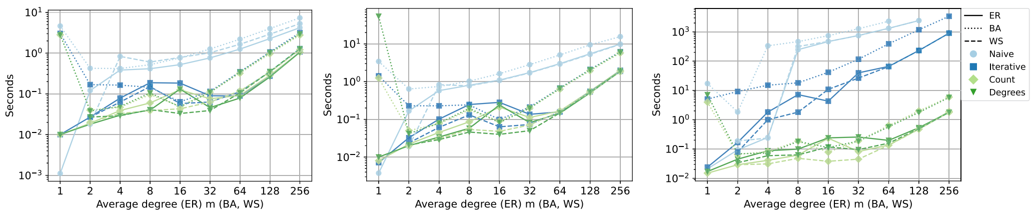
— NAIVE: immediately compute for $d = 5$ (Algorithm 1, no heuristics).
— ITERATIVE: iteratively increase neighborhood radius (Algorithm 2, no heuristics).
— TWIN: ITERATIVE + preprocess twin nodes (no heuristics).
— COUNT: TWIN + the number of nodes and edges heuristic.
— DEGREES: TWIN + the degree distribution heuristic.

All experiments are conducted on a machine with 512 GB RAM (however, memory is never a limiting factor), 128 AMD EPYC 7702 cores at 2.00 GHz, and 256 threads. During the experiments, each run uses one thread, which is not shared with different processes.

6.2 Graph Models

To compare the performance of the different configurations, we first discuss results on different graph models with various network densities. The results can be found in Figure 2 where the runtime is plotted for the three graph models using three different settings (one in each subfigure). The density increases from an average degree (ER), or m, the number of nodes to which each node is initially connected (BA, WS), from 1 for ER and BA, or 2 for WS (very sparse), to 256 (dense). To better show the difference in results over the various graph models, runtime results for each separate graph model can be found in Appendix B, Figure 8.

We first focus on Figure 2(a), which shows the default settings described in Section 6.1. The largest speedup, of several orders of magnitude, is achieved when switching from the NAIVE to the ITERATIVE approach, except for ER graphs with average degree 1. Compared to NAIVE, which still achieves average runtimes of under 10 seconds on all networks, these configurations are in some



(a) Default (b) Twin nodes (c) No cache

Fig. 2. Runtime (vertical axis) of different algorithm configurations (indicated by color) on the three graph models (indicated by line type) with 1,000 nodes, for increasing network density measured either by "Average degree" or "m" (horizontal axis). Each result is averaged over 10 generated graphs.

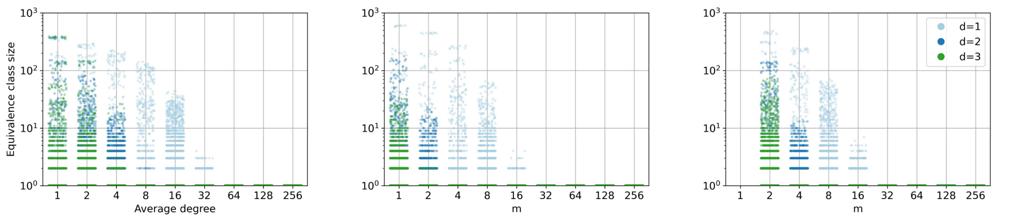
cases 100 times faster. A part of this speedup can be explained by the different radii used by both approaches. In the naive approach, choosing a larger neighborhood radius d results in more cases where expensive canonical labelings have to be computed. The other part, which is discussed in Section 6.2.1, can be explained by the anonymity distribution.

When adding a heuristic to the iterative approach, the overall runtimes decrease slightly. For graphs with small average degrees, a speedup of up to a factor 10 can be observed and for more dense networks, the runtimes observed are very similar. The overall effect of adding these heuristics is that smaller candidate classes are created. In most cases, these smaller equivalence classes will result in a smaller number of already cheap comparisons. However, when there are no candidates and the candidate class is empty, no canonical labeling has to be computed. This has the potential to save significant amounts of time, but for graph models this affects the runtime only slightly. When comparing the heuristics themselves, their achieved runtimes are very similar. While the DEGREES heuristic is more expensive to compute, it also contains more information than COUNT, but this does not result in a significant difference. As a robustness check, runtimes on graph models with 10,000 nodes are included in Appendix B Figure 7. The results show similar differences between the configurations as reported for 1,000 nodes in Figure 2(a).

To assess the effect of the twin node preprocessing step, we have included additional results for the same configurations where the twin node preprocessing step is included in Figure 2(b). Compared to the results without cache, this step in most cases increases the runtimes. We believe that this happens because twin nodes are less common in random graph models.

Next, we assess the importance of using a cache. In Figure 2(c), the runtimes for the graph models are plotted, analogously to Figure 2(a), when no cache is used (see Section 4.2). When comparing the figures, they show that the use of the cache results in a speedup of multiple orders of magnitude for the NAIVE and ITERATIVE configurations, while for the configurations with heuristics only a small speedup is obtained. For NAIVE, some of the most dense instances are not even finished within the time limit of one hour. However, when not using the cache a very large speedup can be obtained by using heuristics. Apparently, the use of a cache or the heuristics both result in major speedups when used separately. When the cache is used, this might already avoid many comparisons that are otherwise avoided by the use of a heuristic.

6.2.1 Performance vs. Anonymity. An important instrument for understanding the performance of the configurations is the anonymity distribution, which is plotted in Figure 3. In this figure, each point represents one equivalence class in one of the graphs. The vertical axis indicates the size of the equivalence class, while the horizontal axis indicates the average degree of the considered graph. This figure shows a decrease in anonymity as a result of increasing density (horizontal



(a) Erdős-Rényi (ER) model (b) Barabási-Albert (BA) model (c) Watts-Strogatz (WS) model

Fig. 3. Anonymity distribution in graph models with different densities measured by either "Average degree" or "m" (horizontal axis) for distances $d = 1, 2$, and 3, indicated by different colors. Each dot represents the size of one equivalence class (vertical axis), which corresponds to the anonymity of all nodes in it. A value of 10^0 denotes an equivalence class containing one (unique) node. Results over ten generated graphs are included for each value for the average degree or m.

axis) and distance d (indicated by different colors). For interpretability of the figures, only results up to distance 3 are included.

In dense graphs, more diverse neighborhoods are possible, which are all unique after an average degree of 64 at distance 1. While the neighborhoods in dense graphs tend to be larger, and therefore the canonical labeling computation more expensive, the increasing uniqueness even at smaller distances means that after the second iteration very few comparisons have to be made. This can explain the large difference in runtime when comparing the NAIVE and ITERATIVE configurations; for the first, 5-neighborhoods of all nodes need to be computed, but at distance 1, a large fraction of the nodes are already unique and significantly less work is required in later iterations. Therefore, ITERATIVE has to compute relatively few canonical labelings of larger neighborhoods.

For all three graph models, the anonymity distribution seems very similar, especially BA and WS. ER does achieve slightly more anonymity overall, which could be a result of the higher density in BA and WS graphs, but also due to the generative process which adds edges randomly rather than based on some other systematic mechanism.

6.3 Empirical Networks

For the second part of the experiments, we look at the performance for the empirical networks listed in Table 1. We have summarized the most important results in Figure 4 and Table 2, the latter containing the runtimes for the five configurations (columns two to six), as explained in Section 6.1 both for $d = 2$ and 5. To summarize the overall performance improvement of the approaches, the seventh column reports for each dataset the speedup obtained when using the

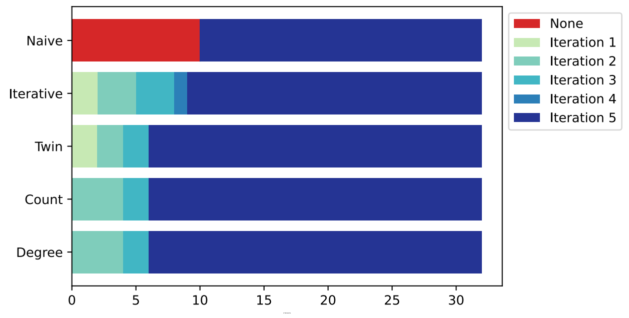


Fig. 4. Last finished iteration (indicated using color) for different configurations (vertical axis).

best performing configuration, being COUNT, DEGREE, or in one case TWIN, compared to NAIVE. The results demonstrate speedups of several orders of magnitude, ranging from 2.50 to 1,154.06. In addition, for more detailed comparison of the configurations, in the text below we report on average speedup values obtained by comparing relevant variants of the algorithm.

Overall, we see similar differences in performance of the configurations as observed for the graph models in Figure 2. Figure 4 illustrates the number of empirical networks on which each configuration finished a certain iteration in the set time limit of three hours. Note that these iterations directly correspond with distances, and $d = 1$ corresponds with 1-neighborhood isomorphism introduced in [20, 31]. In the figure, it can be seen that the ITERATIVE configuration finishes one additional network up to $d = 5$. When using the twin node preprocessing step, three more networks are finished within the time limit. The configurations that use heuristics always finish at least the second iteration, while for other configurations this is not always the case; ITERATIVE and TWIN finish only the first iteration in two cases, and NAIVE none of the iterations in 10 cases. However, note that since NAIVE requires only one iteration, which is the same as the fifth iteration for other configurations, this configuration either finishes all iterations or none. Overall, by using DEGREE or COUNT, we can compute d-k-anonymity with $d \leq 2$ for all included networks, and $d = 3$ for most.

We now turn to more detailed results per network, shown in Table 2, focussing on the case when $d = 5$. When comparing the results to NAIVE without any heuristics, using ITERATIVE resulted in a speedup of multiple orders of magnitude on empirical networks. When accounting for all runtimes reported for the NAIVE configuration, using a heuristic leads to an average speedup of 99.36 and 98.56 for COUNT and DEGREES compared to NAIVE. Compared to TWIN this equals 1.27 and 1.23 respectively which shows that using a heuristic results in a small speedup on the empirical data. This is similar to the results on graph models.

However, adding the twin node step to the iterative approach does result in a significant speedup of 7.97 on average. For most networks, this value is between 1 and 3, but for some networks, being "Bitcoin alpha", "DNC e-mails", "DT interaction" and "Gene fusion" this speedup is much larger. These are all networks with a large fraction of twin nodes skipped, as shown in Table 1, which implies that due to the preprocessing step a large fraction of nodes do not have to be taken into account and no comparisons or canonical labeling computations are required.

Overall, for the empirical networks the use of the iterative approach has the largest effect on the runtime and thereafter the twin node preprocessing step. Adding heuristics results in a small speedup and a shorter runtime on most networks, except for a few.

Now turning to the results for $d = 2$, we note that all networks can be processed, albeit only when heuristics are employed. Similar trends as for $d = 5$ are observed, with heuristics demonstrating speedups of 99.36 and 98.56 on average for COUNT and DEGREE compared to NAIVE. For $d = 2$ the effect of preprocessing twin nodes is smaller and results in a speedup of 1.39 compared to ITERATIVE. At the same time, adding the heuristics results in larger speedups of 14.28 and 21.55 for COUNT and DEGREE compared to TWIN.

6.3.1 Performance vs. Anonymity. In order to further understand the largest differences in performance in Section 6.3, we focus on the anonymity of the nodes in the networks, which (recall from Sections 3 and 4) is equal to the size of the equivalence class a node belongs to. The seventh column of Table 1 shows that the fraction of unique nodes at distance 1 can differ a lot between networks. For a few networks this contains only a small fraction of nodes (e.g., for "DT interaction" and "Network science"). However, for most networks this equals a fraction larger than 0.1 and for most Facebook networks, this is larger than 0.9. This differs from the artificial graph models in Section 6.2 where all nodes were unique at an average degree (or m for BA and WS) of 32 or larger. Thus, unlike for the included dense graph models, for empirical networks comparisons need to be

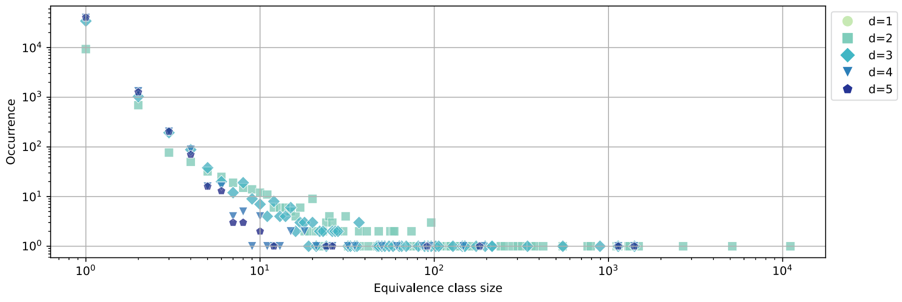
Table 2. Runtime Results for Computing d-k-anonymity for $d = 2$ and $d = 5$ on Empirical Networks, using Five Configurations and a Time Limit of Three Hours

Network	NAIVE	ITERATIVE	TWIN	COUNT	DEGREE	Speedup
			$d = 2$			
Radoslaw email	0.03 (0.01)	0.02 (0.00)	0.02 (0.00)	0.01 (0.00)	0.01 (0.00)	2.5
Primary school	0.06 (0.01)	0.02 (0.00)	0.02 (0.00)	0.02 (0.00)	0.02 (0.00)	3.3
Moreno innov.	0.00 (0.00)	0.00 (0.00)	0.00 (-)	0.00 (-)	0.00 (-)	-
Gene fusion	0.02 (0.00)	0.01 (0.00)	0.00 (-)	0.00 (-)	0.00 (-)	-
Copnet calls	0.06 (0.01)	0.03 (0.00)	0.03 (0.00)	0.02 (0.00)	0.02 (0.00)	3.33
Copnet sms	0.02 (0.01)	0.01 (0.00)	0.00 (0.01)	0.00 (-)	0.00 (-)	-
Copnet FB	0.18 (0.02)	0.04 (0.00)	0.03 (0.00)	0.02 (0.00)	0.02 (0.00)	8.80
FB Reed98	0.78 (0.05)	0.10 (0.00)	0.09 (0.00)	0.07 (0.00)	0.08 (0.00)	10.86
Arenas email	0.15 (0.00)	0.05 (0.00)	0.04 (0.00)	0.03 (0.00)	0.03 (0.00)	-
Network science	0.03 (0.01)	0.03 (0.00)	0.02 (0.00)	0.02 (0.00)	0.02 (0.00)	1.60
FB Simmons81	1.79 (0.15)	0.16 (0.00)	0.14 (0.00)	0.12 (0.01)	0.12 (0.00)	15.41
DNC emails	59.85 (1.58)	19.86 (1.13)	5.72 (0.02)	0.36 (0.01)	0.30 (0.01)	196.89
Moreno health	0.25 (0.00)	0.15 (0.01)	0.14 (0.00)	0.06 (0.00)	0.06 (0.01)	4.17
FB Wellesley22	8.79 (0.28)	0.54 (0.05)	0.50 (0.00)	0.43 (0.01)	0.47 (0.01)	20.26
Bitcoin alpha	48.55 (3.09)	16.09 (0.25)	9.50 (0.03)	0.37 (0.00)	0.40 (0.01)	131.92
GRQC collab	0.90 (0.06)	0.38 (0.01)	0.35 (0.01)	0.21 (0.00)	0.21 (0.00)	4.25
FB Carnegie49	47.46 (1.80)	2.12 (0.06)	1.99 (0.00)	1.73 (0.01)	1.82 (0.02)	27.49
Pajek Erdős	219.27 (54.68)	10.27 (0.19)	6.66 (1.54)	0.21 (0.00)	0.19 (0.01)	1,154.06
DT interaction	251.92 (17.73)	212.15 (11.24)	197.82 (17.19)	4.73 (0.02)	4.03 (0.04)	62.57
DG association	1,295.36 (76.40)	1,085.71 (9.14)	982.96 (10.05)	4.86 (0.02)	2.53 (0.01)	512.00
FB GWU54	140.59 (5.75)	3.99 (0.01)	3.89 (0.02)	3.34 (0.01)	3.44 (0.02)	42.07
Anybeat	-	-	-	4,668.03 (29.54)	4,055.49 (36.46)	-
CE-CX	37.22 (2.98)	3.23 (0.02)	3.13 (0.00)	2.38 (0.01)	2.54 (0.01)	15.63
Astro physics	808.86 (40.43)	66.48 (3.66)	61.32 (0.55)	45.86 (0.28)	46.23 (0.24)	17.64
FB BU10	209.32 (6.86)	5.76 (0.07)	5.50 (0.17)	4.29 (0.02)	4.29 (0.01)	48.84
FB Ullinois	475.07 (14.28)	11.12 (0.22)	10.51 (0.35)	7.94 (0.02)	7.95 (0.08)	59.83
Enron email	-	3,573.90 (109.36)	746.16 (1.74)	254.41 (0.80)	254.19 (0.41)	0.00
FB Penn94	771.28 (38.40)	18.56 (0.51)	17.67 (0.73)	13.33 (0.03)	13.17 (0.03)	58.57
FB wall 2009	48.79 (3.49)	8.33 (0.03)	8.46 (0.05)	2.06 (0.02)	1.93 (0.01)	25.33
Brightkite	2,281.35 (124.25)	408.02 (9.98)	354.75 (15.35)	17.50 (0.07)	17.70 (0.05)	130.36
The marker cafe	-	-	-	624.93 (28.31)	649.92 (25.59)	-
Slashdot zoo	-	7,798.00 (94.65)	7,051.39 (36.36)	453.15 (10.95)	476.96 (80.89)	-
			$d = 5$			
Radoslaw e-mails	0.03 (0.00)	0.03 (0.00)	0.02 (0.00)	0.01 (0.00)	0.02 (0.00)	2.50
Primary school	0.06 (0.00)	0.02 (0.00)	0.02 (0.00)	0.02 (0.00)	0.02 (0.00)	3.33
Moreno innov.	0.01 (0.00)	0.01 (0.00)	0.01 (0.00)	0.00 (-)	0.00 (-)	-
Gene fusion	0.07 (0.00)	0.11 (0.00)	0.01 (0.01)	0.00 (-)	0.00 (-)	-
Copnet calls	0.14 (0.00)	0.05 (0.00)	0.03 (0.00)	0.02 (0.00	0.02 (0.00)	7.78
Copnet sms	0.08 (0.00)	0.02 (0.00)	0.01 (0.00)	0.01 (0.00)	0.01 (0.00)	10.00
Copnet FB	0.35 (0.00)	0.04 (0.00)	0.03 (0.00)	0.02 (0.00)	0.03 (0.01)	15.91
FB Reed98	0.96 (0.01)	0.14 (0.00)	0.10 (0.00)	0.08 (0.00)	0.09 (0.00)	12.26
Arenas email	1.61 (0.00)	0.16 (0.01)	0.05 (0.01)	0.04 (0.00)	0.04 (0.00)	42.37
Network science	0.52 (0.00)	0.37 (0.01)	0.31 (0.00)	0.30 (0.00)	0.31 (0.01)	1.72
FB Simmons81	2.41 (0.01)	0.21 (0.01)	0.15 (0.00)	0.13 (0.01)	0.13 (0.00)	18.57
DNC emails	3,535.10 (156.67)	5,325.36 (379.89)	135.77 (1.50)	129.90 (0.63)	130.74 (1.59)	27.21
Moreno health	2.69 (0.01)	0.17 (0.01)	0.15 (0.00)	0.07 (0.00)	0.06 (0.01)	43.35
FB Wellesley22	12.20 (0.30)	0.54 (0.05)	0.50 (0.01)	0.44 (0.01)	0.48 (0.02)	27.73
Bitcoin alpha	7,973.51 (202.41)	4,212.82 (85.59)	52.54 (0.71)	44.34 (2.41)	43.20 (0.48)	184.58
GRQC collab	1,210.61 (24.30)	222.06 (9.64)	177.83 (0.73)	179.18 (0.80)	179.00 (0.78)	6.76
FB Carnegie49	93.05 (2.94)	3.48 (0.08)	2.22 (0.00)	1.96 (0.01)	2.04 (0.01)	47.57
Pajek Erdős	-	-	2,911.55 (289.22)	3,095.08 (31.53)	3,101.19 (58.17)	-
DT interaction	-	9,949.75 (216.24)	743.97 (17.09)	607.91 (3.58)	609.55 (5.46)	-
DG association	-	-	1,875.30 (24.01)	874.41 (1.38)	878.53 (15.22)	-
FB GWU54	396.43 (4.69)	8.09 (0.04)	4.49 (0.03)	3.95 (0.01)	4.05 (0.02)	100.41
Anybeat	-	-	-	-	-	-
CE-CX	1,381.02 (23.26)	12.37 (0.14)	4.28 (0.01)	3.47 (0.01)	3.62 (0.02)	398.45
Astro physics	-	-	-	-	-	-
FB BU10	1,384.50 (25.45)	14.85 (0.13)	6.85 (0.24)	5.60 (0.03)	5.59 (0.01)	247.85
FB Ullinois	3,009.54 (82.29)	22.85 (0.69)	11.77 (0.37)	9.17 (0.01)	9.19 (0.09)	328.12
Enron email	-	-	-	-	-	-
FB Penn94	9,117.50 (110.58)	55.17 (1.01)	23.80 (1.03)	19.28 (0.03)	19.17 (0.03)	475.66
FB wall 2009	-	-	653.23 (2.27)	638.05 (1.96)	639.09 (2.90)	-
Brightkite	-	-	-	-	-	-
The marker cafe	-	-	-	-	-	-
Slashdot zoo	-	-	-	-	-	-

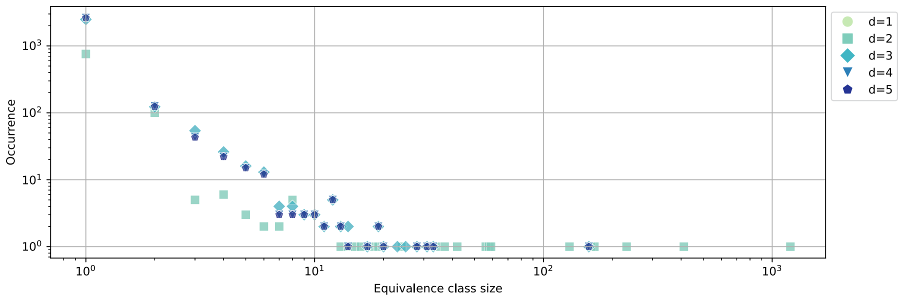
Runtimes and standard deviations reported between parenthesis are measured over five runs. Runs that did not terminate are denoted "-". Speedups reported compare the runtime of the best configuration to NAIVE.

made to determine if nodes are still equivalent at a larger distance which implies that computational effort is still required in later iterations, even in networks with high densities.

In Figure 5, the anonymity distribution is plotted for two networks: "FB wall" and "Bitcoin alpha". Overall, their distributions both appear to resemble a fat-tailed power-law, which interestingly



(a) FB wall



(b) Bitcoin alpha

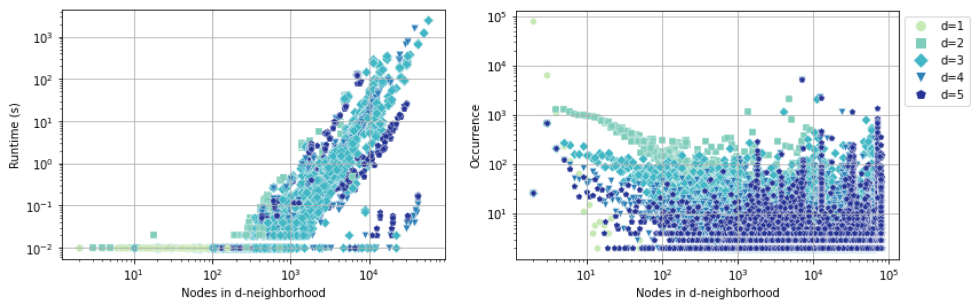
Fig. 5. Anonymity distributions in two empirical networks. Each dot represents the number of occurrences (vertical axis) of that equivalence class size (horizontal axis). This size corresponds to the anonymity of all nodes in it. A value of 10^0 denotes an equivalence class containing one (unique) node.

occurs for all networks that have a large enough number of distinguishable equivalence class sizes. Yet, the anonymity distribution can vary greatly per network.

For the first network, equivalence class sizes from different iterations can be distinguished. This shows that even during the fifth iteration, there are still large equivalence classes with over a thousand nodes. The number of unique nodes, however, increases quickly when the distance equals 2. This increase seems to have the most effect on the overall anonymity. From distance 3 onwards, the anonymity only decreases slightly. For the “Bitcoin alpha” network, a lot of anonymity is lost when increasing to distance 2, and further increasing the distance affects the anonymity distribution only slightly. In this figure, only three iterations can effectively be distinguished. Overall, the most substantial differences in anonymity are observed between $d = 1$ and $d = 2$. This finding may in particular prove relevant for previously proposed approaches such as [20, 31], which explicitly attempt to shed light on (attacks on) privacy in social networks, yet are essentially limited to a possibly less realistic attacker scenario, i.e., a value of $d = 1$.

6.3.2 Twin Nodes and Canonical Labeling. We observed in Section 6.3 that including the twin node preprocessing step can lead to significantly lower runtime on empirical networks. Below, we investigate possible causes.

The most important (and possibly expensive) step of computing d-k-anonymity is the computation of canonical labelings. To understand the runtime of this computation, we plot for all empirical networks the size of the d-neighborhoods of nodes against the time it takes to compute its canonical labeling in Figure 6(a). All results plotted are obtained by the COUNT configuration in the time limit of three hours. This figure shows that for most neighborhoods encountered,



(a) Canonical labeling computation runtime (b) Twin node neighborhood size distribution

Fig. 6. Sizes of d-neighborhoods (horizontal axis) plotted against the runtime to compute its canonical labeling (a) and the occurrence of this size for twin nodes (b). The distance d is indicated by the color.

canonical labeling computation actually takes under a hundredth of a second. However, for larger neighborhoods, starting at 100 nodes, the runtime required can increase significantly. These neighborhood sizes occur after the first iteration. When neighborhoods contain over 10,000 nodes, the computation time can be over 15 minutes.

These nodes can be seen as so-called bottleneck nodes which can immensely impact the runtime of the algorithms and eventually cause it to not compute d-k-anonymity for larger distances in the set time limit. To illustrate this, we included results for the four largest networks in Figure 9 in Appendix C. These figures show that for "Brightkite" and "The Marker Cafe", two networks for which the algorithm did not finish in the given time limit, neighborhoods were encountered for which the canonical labeling computation requires over 15 minutes. At the same time, for "FB Penn94" and "FB wall", the computation takes at most seconds for a single node. For these networks the algorithm does finish all iterations.

In Figure 6(b), the sizes of the d-neighborhoods of the twin nodes that occurs in one of the 32 empirical networks are plotted. This shows that twin nodes can have large neighborhoods, which in turn lead to possibly long canonical labeling computation times. The figure also shows that the number of nodes in the 1-neighborhoods of the twin nodes, which equals their degree plus one, is relatively small; at most 100. When limiting the maximum degree to a small value, such as five which is used in our experiments, we capture a very large fraction of twin nodes.

7 CONCLUSIONS AND FUTURE WORK

In this article, we investigated algorithms for efficient computation of a new tunable measure for network anonymity called d-k-anonymity. In addition to a naive and iterative algorithm, we introduced a preprocessing step and two different heuristics. These methods allow us to measure d-k-anonymity of large empirical networks with tens of thousands of nodes and millions of edges.

To analyze and understand the performance of the proposed methods, we performed experiments on both three well-known graph models and a wide range of empirical networks. On both types of networks, the use of the iterative approach with heuristics resulted in a significant speedup of up to 100 or more for both graph models and empirical networks. The preprocessing step that finds twin nodes, which are by definition structurally equivalent, by itself reduced running time by around 8 times. The used heuristics had a smaller effect and lead to a further performance increase of about 1.25 on average. Our experiments also showed that for graph models with high densities all nodes are unique after $d = 1$ and hence no computational work is required in the later iterations. However, for empirical networks this does not hold and work is still required in later iterations, even for networks with a high density.

While the introduced approaches demonstrated a significant speedup, we saw that the DEGREE configuration did not improve much on COUNT. Hence, we do not expect more complicated heuristics to be effective. However, there are still other possible avenues for further research. First, since computing the canonical labeling of large neighborhoods is expensive but not often required, we expect that the performance can extensively benefit from parallelization of the algorithm. Second, as for a more complicated avenue of research, we expect that an iterative approach to canonical labelings, which reuses results from the previous iteration, could be beneficial. Third, the proposed measures can be extended to protect more complex types of networks that feature link directionality, node and edge attributes, or temporal edges. Another avenue for future work could be to measure the resulting data utility using various anonymization methods and a certain anonymity measure. How to best balance the strictness of the chosen measure for anonymity and resulting data utility may very well be highly dependent on a combination of the empirical network's structure, the chosen network measure (e.g., centrality measures) or network process (e.g., influence or epidemic spread models) that is being applied.

The presented algorithms for computing anonymity measures and experimental findings on anonymity distributions have the potential to fuel the development of techniques to anonymize networks. More broadly, the algorithms introduced in this work pave the way for privacy-aware sharing of sensitive network data.

APPENDICES

A PROOFS

This appendix contains the proof of Theorem 4.2 on twin nodes presented and further discussed in Section 4.4.

PROOF. Given a graph $G = (V, E)$ and twin nodes $v_1, v_2 \in V$, then there exist at least two automorphisms. First, γ such that $\forall_{v \in V} : \gamma(v) = v$. Second, γ' such that $\forall_{v \in V (v \neq v_1, v_2)} : \gamma'(v) = v$, $\gamma'(v_1) = v_2$ and $\gamma'(v_2) = v_1$. Because of the twin node property defined in Definition 3.5 we know that if $\forall_{w\ s.t. \{v_1, w\} \in E} : \{v_2, w\} \in E$. Therefore, $\forall_w : \{\gamma'(v_1), w\} \in E$ also $\{v_1, w\} \in E$. Logically, the same holds for v_2, and hence we can conclude that γ' is a valid automorphism, and given any pair of twin nodes $v_1, v_2 \in V$, the nodes are thus in the same orbit. □

B SUPPLEMENTARY RESULTS: GRAPH MODELS RUNTIME

This appendix contains supplementary figures for Section 6.2. The figures contain the runtimes for graph models with 10,000 nodes, and results per graph model, as reported in Figure 2.

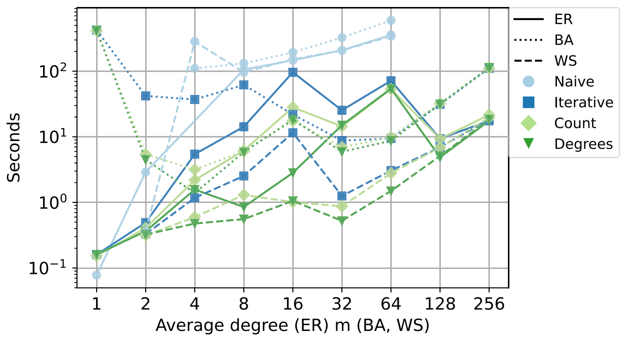
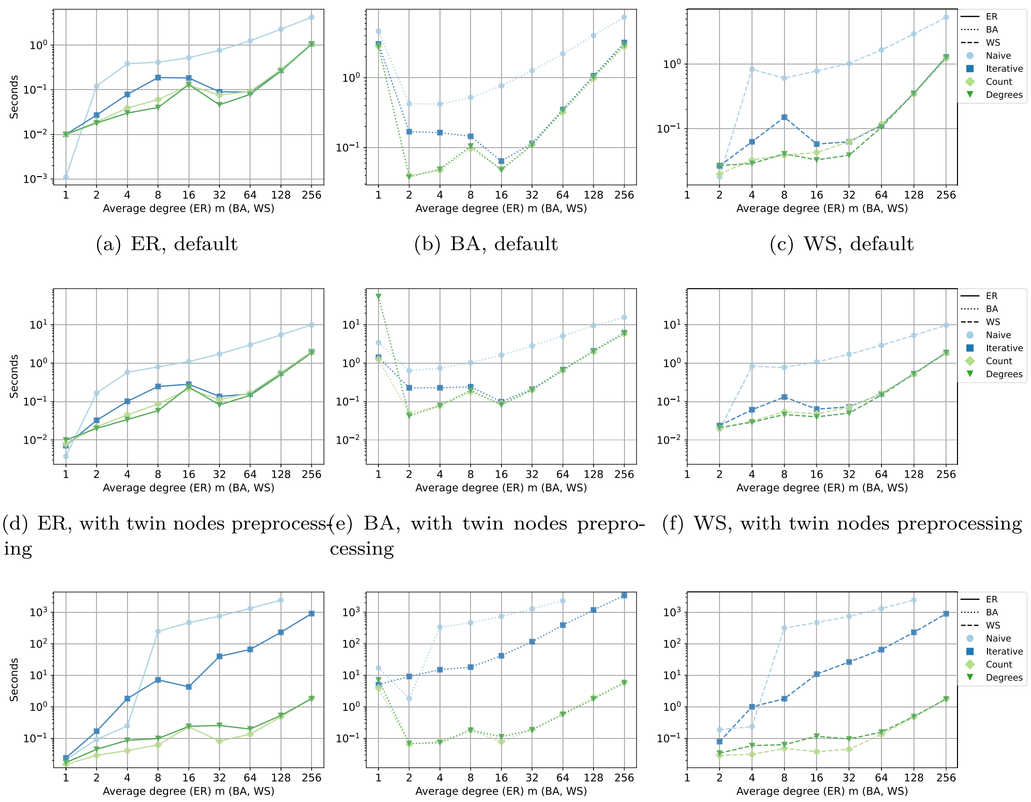


Fig. 7. Runtime (vertical axis) of different algorithm configurations (indicated by color) on the three graph models (indicated by line type), for increasing network density (horizontal axis). Each result is averaged over 10 generated graphs with 10,000 nodes.



(a) ER, default (b) BA, default (c) WS, default

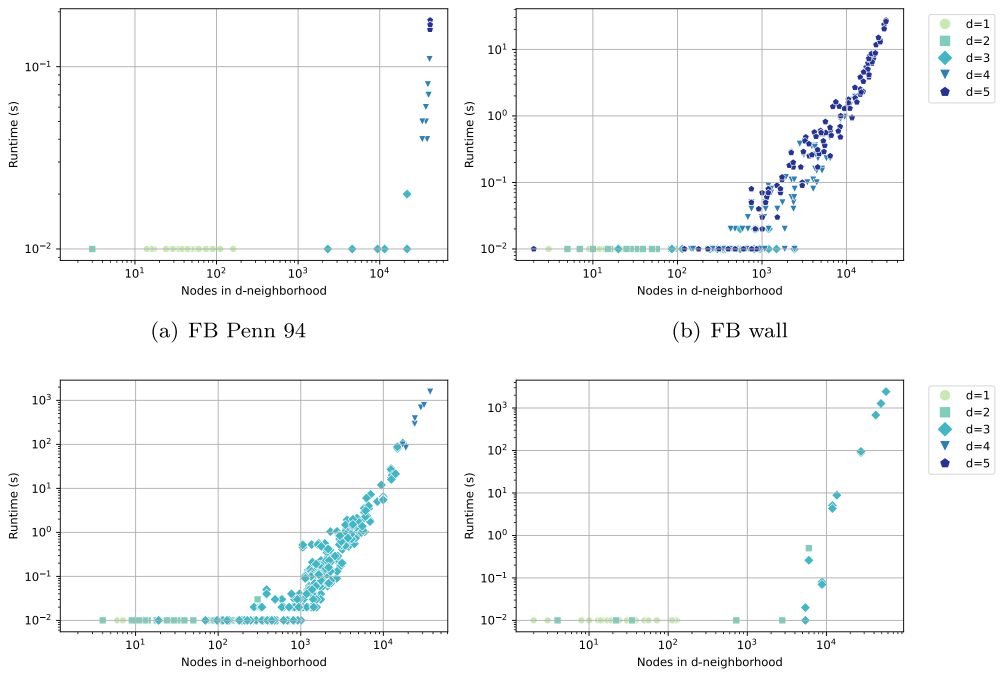
(d) ER, with twin nodes preprocessing (e) BA, with twin nodes preprocessing (f) WS, with twin nodes preprocessing

(g) ER, no cache (h) BA, no cache (i) WS, no cache

Fig. 8. Runtime (vertical axis) of different algorithm configurations (indicated by color) on the three graph models (indicated by line type), for increasing network density (horizontal axis). Each result is averaged over ten generated graphs. Results included are using the default setting (top row), with twin node preprocessing step (middle row) and without cache (bottom row).

C SUPPLEMENTARY RESULTS: CANONICAL LABELING BOTTLENECK NODES

Figure 9 presents supplementary results accompanying Section 6.3.2. The figure is similar to Figure 6(a), but instead of covering all network datasets, presents results for the four largest empirical networks.



(a) FB Penn 94 (b) FB wall

(c) Brightkite (d) The Marker cafe

Fig. 9. Sizes of d-neighborhoods (horizontal axis) plotted against the runtime to compute its canonical labeling (vertical axis) for four specific empirical networks.

ACKNOWLEDGMENTS

We would also like to thank the POPNET team (https://www.popnet.io) for various helpful suggestions and discussions. Additionally, we would like to thank the anonymous reviewers for their valuable comments.

REFERENCES

[1] Asma Azizi, Cesar Montalvo, Baltazar Espinoza, Yun Kang, and Carlos Castillo-Chavez. 2020. Epidemics on networks: Reducing disease transmission using health emergency declarations and peer communication. *Infectious Disease Modelling* 11, 5 (2020), 12–22.

[2] Albert László Barabási. 2016. *Network Science*. Cambridge University Press.

[3] Albert László Barabási and Réka Albert. 1999. Emergence of scaling in random networks. *Science* 286, 5439 (1999), 509–512.

[4] Michał Bojanowski and Rense Corten. 2014. Measuring segregation in social networks. *Social Networks* 39 (2014), 14–32. https://www.sciencedirect.com/science/article/pii/S0378873314000239.

[5] James Cheng, Ada Wai-chee Fu, and Jia Liu. 2010. K-isomorphism: Privacy preserving network publication against structural attacks. In *Proceedings of the 11th ACM SIGMOD International Conference on Management of Data*. 459–470.

[6] Paul Erdős and Alfréd Rényi. 1960. On the evolution of random graphs. *Publication of the Mathematical Institute of the Hungarian Academy of Sciences* 5, 1 (1960), 17–60.

[7] Michael Fire. 2020. Data 4 Good Lab. Retrieved May, 2022 from https://data4goodlab.github.io/MichaelFire/#section3.

[8] Aric Hagberg, Pieter Swart, and Daniel Schult. 2008. *Exploring Network Structure, Dynamics, and Function using NetworkX*. Technical Report. Los Alamos National Lab (LANL), Los Alamos, NM (United States).

[9] Michael Hay, Gerome Miklau, David Jensen, Philipp Weis, and Siddharth Srivastava. 2007. Anonymizing social networks. *Computer Science Department Faculty Publication Series* (2007), 180. https://scholarworks.umass.edu/cs_faculty_pubs/180/.

[10] Anco Hundepool, Josep Domingo-Ferrer, Luisa Franconi, Sarah Giessing, Eric Schulte Nordholt, Keith Spicer, and Peter-Paul De Wolf. 2012. *Statistical Disclosure Control*. Wiley New York. https://www.wiley.com/en-us/Statistical+Disclosure+Control-p-9781118348215.

[11] Shouling Ji, Prateek Mittal, and Raheem Beyah. 2016. Graph data anonymization, de-anonymization attacks, and de-anonymizability quantification: A survey. *IEEE Communications Surveys & Tutorials* 19, 2 (2016), 1305–1326.

[12] Jérôme Kunegis. 2013. Konect: The Koblenz network collection. In *Proceedings of the 22nd International Conference on World Wide Web*. 1343–1350.

[13] Jure Leskovec and Andrej Krevl. 2014. SNAP Datasets: Stanford Large Network Dataset Collection. Retrieved May, 2022 from http://snap.stanford.edu/data.

[14] Kun Liu and Evimaria Terzi. 2008. Towards identity anonymization on graphs. In *Proceedings of the 4th ACM SIGMOD International Conference on Management of Data*. 93–106.

[15] Mark van der Loo. 2022. *Topological Anonymity in Networks*. Discussion Paper. Statistics Netherlands, The Hague. Retrieved from https://www.cbs.nl/en-gb/background/2022/17/topological-anonymity-in-networks.

[16] Priya Mahadevan, Dmitri Krioukov, Kevin Fall, and Amin Vahdat. 2006. Systematic topology analysis and generation using degree correlations. *ACM SIGCOMM Computer Communication Review* 36, 4 (2006), 135–146.

[17] Brendan D. McKay and Adolfo Piperno. 2014. Practical graph isomorphism, II. *Journal of Symbolic Computation* 60 (2014), 94–112. https://www.sciencedirect.com/science/article/pii/S0747717113001193.

[18] Adam Meyerson and Ryan Williams. 2004. On the complexity of optimal k-anonymity. In *Proceedings of the 23rd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*. 223–228.

[19] Hyoungmin Park and Kyuseok Shim. 2007. Approximate algorithms for k-anonymity. In *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data*. 67–78.

[20] Daniele Romanini, Sune Lehmann, and Mikko Kivelä. 2021. Privacy and uniqueness of neighborhoods in social networks. Scientific Reports 11, 1 (2021), 20104.

[21] Ryan A. Rossi and Nesreen K. Ahmed. 2015. The network data repository with interactive graph analytics and visualization. In *Proceedings of the AAAI*. Retrieved May, 2022 from https://networkrepository.com.

[22] Alessandra Sala, Xiaohan Zhao, Christo Wilson, Haitao Zheng, and Ben Y. Zhao. 2011. Sharing graphs using differentially private graph models. In *Proceedings of the 11th ACM SIGCOMM Internet Measurement Conference*. 81–98.

[23] Pierangela Samarati. 2001. Protecting respondents identities in microdata release. *IEEE Transactions on Knowledge and Data Engineering* 13, 6 (2001), 1010–1027.

[24] Piotr Sapiezynski, Arkadiusz Stopczynski, David D. Lassen, and Sune L. Jørgensen. 2019. The Copenhagen Networks Study Interaction Data. Figshare. Retrieved May, 2022 from https://doi.org/10.6084/m9.figshare.7267433.v1.

[25] Sociopatterns. 2021. Sociopatterns: Datasets. Retrieved May, 2022 from http://www.sociopatterns.org/datasets/.

[26] Latanya Sweeney. 2002. Achieving k-anonymity privacy protection using generalization and suppression. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 10, 05 (2002), 571–588.

[27] Duncan J. Watts and Steven H. Strogatz. 1998. Collective dynamics of "small-world" networks. *Nature* 393, 6684 (1998), 440–442.

[28] Leon Willenborg and Ton de Waal. 2001. *Elements of Statistical Disclosure Control*, Vol. 155. Springer Science & Business Media.

[29] Raymond Chi-Wing Wong, Jiuyong Li, Ada Wai-Chee Fu, and Ke Wang. 2006. (α, k)-anonymity: An enhanced k-anonymity model for privacy preserving data publishing. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 754–759.

[30] Wentao Wu, Yanghua Xiao, Wei Wang, Zhenying He, and Zhihui Wang. 2010. K-symmetry model for identity anonymization in social networks. In *Proceedings of the 13th International Conference on Extending Database Technology*. 111–122.

[31] Bin Zhou and Jian Pei. 2008. Preserving privacy in social networks against neighborhood attacks. In *Proceedings of the 24th IEEE International Conference on Data Engineering*. 506–515.

[32] Marinka Zitnik, Rok Sosič, Sagar Maheshwari, and Jure Leskovec. 2018. BioSNAP Datasets: Stanford. Biomedical Network Dataset Collection. Retrieved May, 2022 from http://snap.stanford.edu/biodata.

[33] Lei Zou, Lei Chen, and M. Tamer özsu. 2009. K-automorphism: A general framework for privacy preserving network publication. In *Proceedings of the of the 35th VLDB Endowment*. 946–957.

Received 28 July 2022; revised 4 April 2023; accepted 3 May 2023