Technische Universität Berlin
Fakultät 4 - Elektrotechnik und Informatik
Institut für Softwaretechnik und Theoretische Informatik
Fachgebiet Algorithmik und Komplexitätstheorie

Bachelor Thesis

Graph Degree Anonymization: Lower Bounds and Heuristics

Author:
Clemens Hoffmann
(332772)

Supervisors:
Prof. Dr. Rolf Niedermeier
Dr. Sepp Hartung
André Nichterlein

March 15, 2014

Abstract. Due to the large interest in analyzing social network structures for scientific and market researches, the anonymization of data sets is of large interest in recent research. Given a simple graph G, the NP-hard k-degree anonymization problem asks for a supergraph G' with a minimum amount of edge additions, such that each vertex degree occurs at least k times. Focusing power law distributed social network graphs, we present various polynomial-time heuristics for k-degree anonymization, based on the dynamic programming algorithm of Liu and Terzi. All the graph degree anonymization heuristics work in two phases: Phase 1 is to compute and k-anonymize the degree sequence of the input graph. In Phase 2, the k-anonymous degree sequence is realized as a supergraph of the input graph. While the heuristic of Liu and Terzi has been shown to be optimal if the solution is very large, we examine its usability for rather small real-world instances. As we were not able to k-anonymize any of the degree sequences of the social network graphs used within this work such that realization is possible in Phase 2, we modified the algorithm of Liu and Terzi in an algorithm engineering process in several steps. Furthermore, we introduce several degree sequence realization heuristics which solve Phase 2 of the graph anonymization process and compare these heuristics regarding to effectiveness and running-time. Using these algorithms, we were able to optimally k-degree-anonymize many large social network graphs with > 100.000 vertices and $k = 200$ in less than an hour.

Abstrakt. Datensätze sozialer Netzwerke sind als Quelle wissenschaftlicher Forschungen sowie Marktforschungen von großer Bedeutung. Um die Privatheit der Nutzer zu schützen oder Datenschutzrichtlinien zu erfüllen, ist die Anonymisierung solcher Datensätze notwendig und Gegenstand aktueller Forschungen. Die vorliegende Arbeit beschäftigt sich mit der k-Grad-Anonymisierung von Graphen sozialer Netzwerke, die in der Regel potenzverteilt sind. Gegeben ein einfacher Graph G und ein Parameter k, ist nach einem Supergraphen von G gefragt, in dem jeder Knotengrad mindestens k mal auftritt. Die in dieser Arbeit behandelten Algorithmen zur Lösung dieses Problems basieren auf einem dynamischen Zwei-Phasen-Algorithmus von Liu und Terzi. In Phase 1 wird die zum Graph gehörende Gradsequenz k-anonymisiert, so dass jeder Grad mindestens k mal in der anonymisierten Sequenz auftritt. In der zweiten Phase wird die anonymisierte Sequenz als Supergraph von G realisiert. Da die in Phase 1 mit dem Algorithmus von Liu und Terzi errechneten k-anonymen Gradsequenzen für die Graphen aus dem verwendeten Test-Set nicht realisierbar sind, wird dieser Algorithmus in einem Algorithm Engineering Prozess erweitert, so dass die Wahrscheinlichkeit für die Realisierbarkeit der berechneten Lösungssequenzen in Phase 2 zunimmt. Darüber hinaus werden für Phase 2 verschiedene Heuristiken zur Realisierung der k-anonymen Gradsequenz als Supergraph des Eingabegraphen

vorgestellt und anhand von Effektivität und Laufzeit verglichen. Mit Hilfe dieser Algorithmen konnten einige sehr große Instanzen mit > 100.000 Knoten und $k = 200$ innerhalb von einer Stunde optimal gelöst werden.

Contents

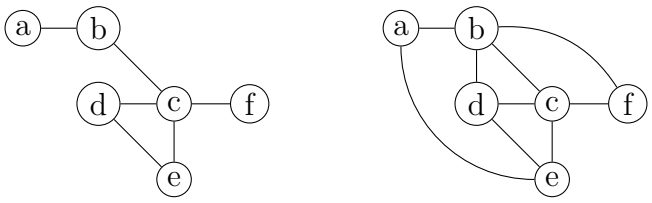
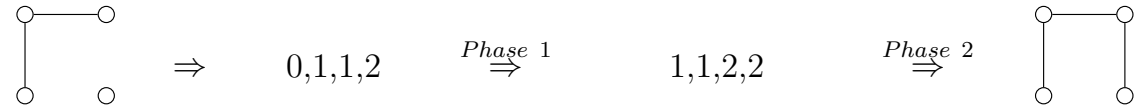


Figure 1: Graph G_0 and a 2-anonymous Graph G_0' with a minimum amount of edge-additions.

1 Introduction

1.1 Motivation

Nowadays, communication and knowledge platforms such as social networks are playing a more and more important role. Such networks are used by more than a billion people to communicate, retrieve information and share content [Fac13]. They supplement or even replace conventional forms of communication. As social networks contain a lot of information about its users, the analysis of implicitly and explicitly contained information is quite interesting for research. For instance, references between users and additional information such as interests, behavior and trends can be analyzed. This means that data from social networks become a more and more important source for scientific and market researches. In order to comply with data privacy laws and to protect the privacy of each user, exported data sets often need to be anonymized before being handed out or processed. There are three different privacy categories: identity disclosure prevents an adversary to reveal the real identity of members within a network. Link disclosure hides sensitive relationships between members and content disclosure hides information associated with each member. Within this work, we focus on identity disclosure in terms of k-anonymity. The subject of k-anonymity is to modify the original data, such that each entity cannot be distinguished from at least $k-1$ other entities within the data while losing as less information as possible. Within this work, social networks are modeled as undirected graphs. Its vertices represent network members and its edges represent social relationships between those.
Backstrom et al. [BDK11] have shown that renaming all members within a network is not sufficient in order to assure that the identity of a member cannot be revealed. With additional knowledge such as the number of relationships between certain members, their real identity can be revealed. Furthermore, Hay et al. [Hay+07] demonstrated that structural analogies like the number of relationships of a member determine whether its identity can be revealed with additional knowledge. For instance, if there is a vertex with a unique amount of links, one can easily determine its real identity when the amount of its links is known. In order to prevent an attacker from revealing the identity of a vertex, we work on the subject of *degree anonymization*. A network is k-degree-anonymous if each vertex degree occurs at least k times. Although this process does not provide perfect privacy (e.g. the structure of a vertex' neighborhood can be unique), it is subject of active researches and forms a fair trade-off between privacy and practicability [LT08].



graph G and $k = 2$ | initial degree sequence D | k-anonymized degree sequence D' | realization of D' in G

Figure 2: A simple example for the 2-Phase heuristic offered by [LT08]. Phase 1: Anonymization of the degree sequence D of the input graph G such that each resulting number occurs at least k times. Phase 2: Realization of the k-anonymized degree sequence D' as a super-graph of G.

There are various strategies to provide degree anonymity, such as removing vertices or edges. Within this work, we focus on the strategy to add as few edges as possible in order to transform the input graph to a k-degree anonymous graph, which has been introduced by Liu and Terzi [LT08]. Figure 1 illustrates a Graph G_0 and a 2-anonymous supergraph G'_0, which is the result of such a graph degree anonymization process. This problem of k-degree anonymity is known to be NP-hard but can be solved in polynomial time if the solution is very large [Har+13]. However, the solution size of real-world instances is usually rather small. Examining the usability of the algorithm of Liu and Terzi for power-law distributed social network graphs with rather small solutions, we recognized that none of our test graphs could be k-degree anonymized. In the following, we present various enhancements and modifications of this algorithms which increase the chances on successfully solving the k-degree-anonymization problem for power-law distributed graphs.

1.2 Content of this Work

The graph k-degree anonymity problem introduced by Liu and Terzi is defined as follows:

Graph k-degree anonymity
Input: A graph $G = (V, E)$, a positive integer $k \leq |V|$ and a positive integer s.
Question: Is there a graph $G' = (V, E')$ with $E' = E \cup S$ and $|S| = 2s$ such that the degree of each vertex $v \in V$ occurs at least k times?

Figure 2 illustrates the concept of k-degree anonymizing a graph by transferring the graph problem into a number problem and realizing the computed solution within the graph. *Phase 1* of this procedure is to k-anonymize the *degree sequence* of the input graph, which stores the degree of each vertex, with minimal cost. This procedure is covered in Section2. Subsequently, the resulting k-anonymous degree sequence is realized as a supergraph of the input graph in *Phase 2*, which is covered in Section4 and Section5. For both phases, we present various strategies and implementations. In order to solve Phase 1, we first introduce the dynamic programming algorithm of Liu and Terzi [LT08] which computes a k-anonymous degree sequence in polynomial-time. As our experiments

revealed that the k-anonymous solution degree sequences of our test instances computed by this algorithm could not be realized as supergraphs of the input graph, we present a slight modification of this heuristic which is supposed to increase the likelihood of realizability in Section 2.2. Furthermore, we present the characterization of Erdős and Gallai [EG60] in Section 2.3, which a necessary constraint for the realizability of degree sequences. As the running time and the amount of solutions of these heuristics depend on the size of the initial degree sequence, we introduce data reduction rules in Section 3. Since our algorithms were ineffective for all tested social network instances, we bring up another approach for Phase 1 in Section 4. Instead of working on degree sequences, we present the concept of computation and anonymization of *degree block sequences*. This concept has been introduced by Hartung et al. [HHN14]. In Section 5, we also introduce various strategies for realizing k-anonymous degree sequences and degree block sequences, which relates to Phase 2. In Section 6, we finally sum up the results of our experiments and compare the miscellaneous realization strategies.

1.3 Related Work

The subject of anonymizing data sets is of large interest in recent research. The concept of k-anonymity has been introduced by Samarati and Sweeney [SS98] for tables as a protection against linking external data against so called *quasi-identifiers* which were believed to be anonymous. Quasi identifiers are attributed such as postal code or birth date which can be used for exploiting the identity of an entity. Instead of removing fields which directly identify any individual such as name, social security number or phone number, a table has been called k-anonymous if each set of quasi-identifiers can be found at least k times within the table. This subject has further been studied by Sweeney [Swe02]. Meyerson and Williams [MW04] proofed that making a relation k-anonymous by removing as few entities as possible is NP-hard. Similar concepts have been introduced for providing anonymity for social networks modeled as graphs. Backstrom et al. [BDK11] have shown that renaming all vertices within a graph is not sufficient. With additional knowledge such as the number of relationships between certain members, their real identity can be revealed. Furthermore, Hay et al. [Hay+07] demonstrated that structural analogies within the graph determine whether its identity can be revealed with additional knowledge. Thus, [Hay+07] introduced the concept of k-candidate anonymization, meaning that for each structural query over a graph, there exist at least k vertices which match this query. The concept of making a graph k-anonymous such that each vertex degree occurs at least k times has been introduced by Liu and Terzi [LT08] as k-degree anonymity. This concept is less strong than the proposed definition of k-candidate anonymity, as the structure of vertices' neighborhoods may still be unique. Thus, the graph-anonymization problem has been defined that given a graph G, it asks for a k-degree anonymous graph G' such that the number of graph-modifications is minimal. Additionally, [LT08] provided simple and efficient polynomial-time heuristic algorithms which are based on the anonymization and realization of degree sequences. Hartung et al. [Har+13] proved that this heuristic is optimal if the amount of edges to add is larger than Δ^4. Bredereck et al. [Bre+13] introduced

the problem of making a graph k-degree-anonymous by removing as few vertices as possible from the input graph. This problem has also proved to be NP-hard even for trees. However, the number of vertices to delete from a power law distributed graph has been shown to be rather low.

1.4 Preliminaries

Throughout this work we consider simple undirected graphs without self-loops and multi-arcs. Formally, a graph $G = (V, E)$ consists of the edge set E and the vertex set $V = \{v_1, \ldots, v_n\}$. Unless explicitly defined otherwise, n denotes the number of vertices. The *vertex degree* of a vertex $u \in V$ is the amount of vertices which are adjacent to u and is denoted by $\texttt{deg}(u)$. The maximal vertex degree within a graph is denoted by Δ.
A *degree sequence* $D = (d_1, \ldots, d_n)$ is a sequence of positive integers. A degree sequence D_G is referred to as the *corresponding* degree sequence to G such that there is a bijective mapping between the entries in D_G and the degrees of the vertices in G. We omit subscripts if the graph is clear from the context. A degree sequence D' is *derived* from D if $d'_i \geq d_i$ for all $1 \leq i \leq n$. We denote a degree sequence D' as realizable if and only if there is a graph G' such that there is a bijective mapping between the degrees of the vertices of G' and the entries of D'. A degree sequence D' is realizable within a graph G if the resulting graph G' is a supergraph of G. A *difference sequence* $D^\Delta = D' - D = (d'_1 - d_1, \ldots, d'_n - d_n)$ results from subtracting two degree sequences D' and D, whereas $d'_i \geq d_i$ for all $i \in \{1, \ldots, n\}$. An entry $d_i^\Delta \in D^\Delta$ denotes the amount of edges which have to be added to v_i, such that $\texttt{deg}(v_i)$ equals d'_i.
The *distance* between two degree sequences D' and D is the sum over all entries of the difference sequence $D' - D$ and denotes the anonymization cost for an input degree sequence D and the solution degree sequence D'. It is denoted by $\texttt{distance}(D', D)$. For $0 \leq p \leq q \leq n$ and $t \geq d_b$, the function $\text{cost}(D, p, q, t)$ computes the distance between the degree sequence $(d'_p = t, \ldots, d'_q = t)$ and the degree sequence $(d_p, \ldots, d_q)$. The amount of *affected vertices* in D' is the Hamming distance between D' and D and is equal to the amount of entries $d'_i > 0$ within D^Δ. The *maximum raise* within two degree sequences D' and D is a largest entry in $D' - D$.
A graph G is k-anonymous if for each vertex degree $\texttt{deg}(v_i)$, $i \in \{1, \ldots, n\}$, there are at least $k - 1$ other vertices v_j such that $\texttt{deg}(v_j)$=$\texttt{deg}(v_i)$. The term of anonymity can also be applied to degree sequences: A degree sequence D is k-anonymous if the value of each entry d_i occurs at least k times within the degree sequence.

2 Computation of Realizable k-anonymous Degree Sequences

The procedure of making a graph k-anonymous is performed in two phases. This section covers Phase 1 of the graph anonymization procedure which is to compute a k-anonymous degree sequence from the input graph as illustrated in Figure 1. We present various

polynomial-time algorithms which compute k-anonymous degree sequence of an input degree sequence. Regarding the graph degree anonymization process, the computed solution degree sequence is optimal if it can be realized within the input graph in Phase 2. However, the distance between the input degree sequence and the k-anonymous solution degree sequence forms a lower bound for the graph k-degree anonymization. The first algorithm presented in Section 2.1 computes a k-anonymous degree sequence with minimal cost, that is the distance between the initial degree sequence and the k-anonymous solution degree sequence is minimal and even. This is a necessary constraint as adding an edge to the graph increases the degree of two vertices by one. Our tests revealed that for each of our social network instances, the computed solution degree sequence could not be realized as a supergraph in Phase 2. The most obvious reason for the lack of realizability is that these instances are power law distributed, e.g. there are very few vertices with a relatively large degree and many vertices with a rather low degree. We observed that a lot of edges have to be added between the vertices with a unique high degree, whereas the vertices with low degrees remain unchanged. Hence, the amount of affected vertices is often smaller than the maximum raise and thus the solution degree sequence cannot be realized. We enhance the algorithm by an additional constraint which ensures that the amount of affected vertices is larger than the maximum raise within the solution degree sequence. An algorithm which implements this constraint is introduced in Section 2.2. As the solution degree sequences computed by this algorithm were not realizable too, we adopt the characterization of Erdős and Gallai [EG60] which checks if a degree sequence is realizable. We show how this characterization can be integrated into Phase 1 of the anonymization process. In Section 2.3, we introduce a modification for this characterization, which takes realizability within the input graph into account.

2.1 Computing k-anonymous Degree Sequences

Before computing a k-anonymous degree sequence as part of the graph degree anonymization process, we need to compute the ascending ordered initial degree sequence D of the input graph G by counting the degree for each vertex. The decision problem of k-anonymizing a degree sequences is defined as follows:

k-Degree Sequence Anonymity (k-DSA)
Input: Two positive integers k and s, and a degree sequence $D = (d_1, \ldots, d_n)$ with $d_1 \leq d_2 \leq \ldots \leq d_n$.
Question: Is there a k-anonymous degree sequence $D' = (d'_1, \ldots, d'_n)$ with $d_i \leq d'_i$ for all $1 \leq i \leq n$ such that $\sum_{i=1}^{n} d'_i - d_i = 2s$?

Obviously, we can find a minimal distance k-anonymous degree sequence D' by computing k-DSA for increasing $s \in \{0, \ldots, n^2\}$, until an algorithm for k-DSA reports (D, k, s) to be a `YES`-instance. The cost of anonymization is the distance between D' and D, which is exactly $2s$. We present *Algorithm 1*, which is a dynamic programming algorithm developed by Hartung et al. [Har+13] because it forms the basis of *Algorithm 2* introduced in Section 2.2.

Algorithm 1: Summary. The algorithm uses a two-dimensional boolean table T with $n \cdot (2s+1)$ entries, each initialized with `false`. An entry $T[i,j], 1 \leq i \leq n, 0 \leq j \leq 2s$ is `true` if and only if the subsequence $(d_1 \ldots, d_i)$ can be transformed into a k-anonymous subsequence $(d'_1, \ldots, d'_i)$ such that $\sum_{t=1}^{i} d'_t - d_t = j$. Thus, if $T[n,j]$ is `true` for an even j, then D is transformable into a k-anonymous degree sequence D' such that the difference between D' and D is exactly j. Having computed T once, D' can be computed by a backtracking algorithm from T. The basic idea of the dynamic programming algorithm is to fragment the input degree sequence into k-anonymous subsequences of length $< 2k$. If an entry of the degree sequence is part of a k-anonymous subsequence with length $\geq 2k$, then this subsequence can be divided into several k-anonymous subsequences with length $< 2k$.

Algorithm 1: Computation of the table. For $i < k$, set $T[i,j]$ to `false`, as there is no k-anonymous degree sequence with less than k elements. For $k \leq i < 2 \cdot k$, set $T[i,j]$ to `true` if and only if there is a $t \in \{d_i, \ldots, \Delta^2\}$, such that the cost of setting the first i entries in D to t is exactly j. In this case, $(d'_1 = t, \ldots, d'_i = t)$ forms the corresponding solution degree sequence. For $i \geq 2k$, set $T[i,j]$ to `true` if and only if there is a $t \in \{d_i, \ldots, \Delta^2\}$ and an $l \in \{k, \ldots, 2k-1\}$, such that $T[i-l, j-\text{cost}(D, i-l+1, i, t)] =$ `true`. In this case, $(d'_{i-l+1} = t, \ldots, d'_i = t)$ forms the corresponding solution degree subsequence. Then, the total cost of making $(d_1, \ldots, d_i)$ k-anonymous is exactly j then. If $T[n,j] =$ `true` for even j, then there is a k-anonymous degree sequence D' with $d'_i \geq d_i$ for all $i \in \{1, \ldots, n\}$ which might be realizable. The dynamic programming algorithm terminates and D' is computed by a backtracking algorithm based on the entries in T. Starting off with $i := n$, the algorithm searches a $t \in \{d_i, \ldots, \Delta^2\}$ and an $l \in \{k, \ldots, 2k-1\}$, such that $T[i-l, j-\texttt{cost}(D, i-l+1, i, t)]$ is `true`. Note that during each step, there might be more than one such t and l which can lead to different solution degree sequences. A combination of t and l is heuristically selected during each step of the backtracking. The corresponding subsequence is $(d'_{i-l+1} = t, \ldots, d'_i = t)$. Set i to $i-l$ and j to $j - \texttt{cost}(D, i-l+1, i, t)$ and repeat this procedure while $i > k$.

Lemma 1. Algorithm 1 solves k-Degree Sequence Anonymity in $O(nsk\Delta)$ time [Har+13].

Limitations. Our experiments with power-law distributed social network graphs revealed that the k-anonymous degree sequences computed by Algorithm 1 cannot be realized within the input graphs. One reason for this is that power law graphs have many vertices with a low degree and very few vertices with a relatively high and unique degree. In order to make the degree sequence of a power law graph k-anonymous, a few entries have to be raised by a relatively huge amount, whereas most other vertices with a low or average degree remain unchanged. Hence, we need to add a lot of edges to a few vertices, whereas there are not enough other vertices that these edges can be made adjacent to. Figure 3 illustrates Graph G_1 demonstrating this issue for $k = 2$. The initial degree sequence of G_1 is $D_1 = (1,1,1,3)$ and the computed degree sequence is $D'_1 = (1,1,3,3)$. This solution is not realizable because it is not possible to raise the

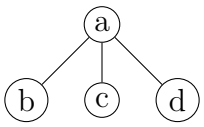


Figure 3: Graph $G_1 = (V_1, E_1)$, $D(G_1) = (1, 1, 1, 3)$

degree of one vertex from 1 to 3 without raising the degree of at least two other vertices. In general, if the maximum raise is larger than the amount of affected vertices within a solution degree sequence D', then D' cannot be realized. Hence, the algorithm is not suitable as part of the graph degree anonymization process. This issue led to adding the requirement that the amount of affected vertices is larger than the maximum raise to the k-Degree Sequence Anonymity problem.

2.2 Considering the Maximum Degree Raise

We introduce k-Degree Sequence Anonymity 2 which is a slight modification of k-Degree Sequence Anonymity. It ensures that the amount of affected vertices within a computed k-anonymous degree sequence is larger than the maximum raise.

k-Degree Sequence Anonymity 2(k-DSA2)
Input: Two positive integers k and s, and a degree sequence $D = (d_1, \ldots, d_n)$ with $d_1 \leq d_2 \leq \ldots \leq d_n$.
Question: Is there a k-anonymous degree sequence $D' = (d'_1, \ldots, d'_n)$ with $d_i \leq d'_i$ for all $1 \leq i \leq n$ such that $\sum_{i=1}^{n} d'_i - d_i = 2s$ and the amount of affected vertices is larger than the maximum raise within D' ?

By slightly modifying Algorithm 1, we can prevent the algorithm from computing k-anonymous degree sequences whose amount of affected vertices is lower than the maximum degree raise. *Algorithm 2* is an enhancement of Algorithm 1, which solves k-Degree Sequence Anonymity 2 heuristically, meaning that computed solutions might not be optimal regarding anonymization cost.

Algorithm 2: Concept. Algorithm 2 uses tables T_y with $n \cdot (2s + 1) \cdot n$ entries each, where $T_y[i, j, x]$ is true if and only if there is a degree sequence $(d'_1, \ldots, d'_i)$ such that the anonymization cost is exactly j, the amount of affected vertices is at least x and the maximum raise is less than y. If an entry $T_y[n, j, x]$ is true for an even j, then there is such a degree sequence D' which might be realizable and the algorithm terminates.

Algorithm 2: Computation of the table. The computation itself is similar to the computation of the table within Algorithm 1. Within this paragraph, the function affectedVertices(D, p, q, t) computes the amount of affected vertices for a degree sequence D and two positive integers $1 \leq p \leq q \leq n$, when the entries $\{d_p, \ldots, d_q\}$ are raised to t. For each $y \in \{1, \ldots, \Delta^2\}$ a new table T_y is created. We are not computing more than Δ^2 tables as there is no entry larger than Δ^2 in the solution degree sequence [HHN14].

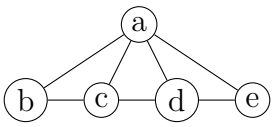


Figure 4: Graph $G_2 = (V_2, E_2)$, $D_2 = (2, 2, 3, 3, 4)$, $D_2' = (2, 2, 4, 4, 4)$

Hence, the maximum raise within D' is Δ^2. Within a table T_y, we iterate over each $j \in \{0, \ldots, 2s\}$ and $x \in \{0, n\}$. For $0 \leq i < k$ we set $T_y[i, j, x]$ to `false` as there is no k-anonymous degree sequence containing less than k entries. For $k \leq i < 2k$ we set $T_y[i, j, x]$ to `true` if and only if there is a $t \in \{d_i, \ldots, \Delta^2\}$, such that the cost of setting the first i entries in D is exactly j, whereas the amount of affected vertices is a least x and the maximum raise is less than y. For $i \geq 2k$, set $T[i, j, x]$ to `true` if and only if there is a $t \in \{d_i, \ldots, \Delta^2\}$ and an $l \in \{k, \ldots, 2k-1\}$, such that $T[i-l, j-cost(D, i-l+1, i, t), x - \text{affectedVertices}(D, i-l+1, i, t)] =$ `true` and $\max(t - d_{i-l+1}, \ldots, t - d_i) < y$. If $T_y[n, j, x] =$ `true` for an even j, a solution degree sequence D' can be generated by a backtracking algorithm similar to Algorithm 1, such that the amount of affected vertices is larger than the maximum degree raise and the total cost of anonymization is even. Note that this algorithm might not compute optimal results reagrding to anonymization cost, as it minimizes the maximum degree raise while keeping anonymization cost as low as possible. This means that there still a solution degree sequence with a larger maximum raise might provide lower anonymization cost in rare cases.

Lemma 2. Algorithm 2 runs in $O(n^2 sk\Delta^4)$ time.

Proof. For each table $T_y, y \in \{1, \ldots, \Delta^2\}$, we iterate over at most $2s \cdot n^2$ entries. For each entry, we need perform at most $k \cdot \Delta^2$ lookups. This results in a running time of $O(n^2 s\Delta^4 k)$. □

Our experiments revealed that computing k-anonymous degree sequences whose number of affected vertices is larger than the maximum raise did not improve the realizability of the computed k-anonymous degree sequences. Besides the huge increase of running time and memory consumption, we could not realize the few instances that could be processed by Algorithm 2 in decent time. The main issue for the lack of realizability is illustrated by graph G_2 shown in Figure 4. The initial degree sequence is $D_2 = (2, 2, 3, 3, 4)$ and for $k = 2$ the solution degree sequence computed by Algorithm 2 is $D_2' = (2, 2, 4, 4, 4)$. The amount of affected vertices is 2 which is larger than the maximum raise which is 1. Besides the fact that both vertices with degree 3 are already adjacent, D_2' cannot be realized even if we do not claim G_2' to be a supergraph of G_2. This shows that the additional property for the amount of affected vertices and the maximum raise within a computed solution degree sequence does not guarantee that this solution degree sequence is realizable. However, the lower bounds computed by Algorithm 2 was larger than the ones computed by Algorithm 1. Our next step in the algorithm engineering process is to integrate the characterization of Erdős an Gallai [EG60] into the degree sequence anonymization process.

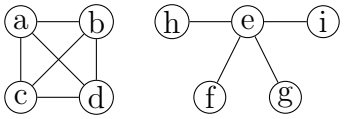


Figure 5: Graph $G_3 = (V_3, E_3)$, $D_3 = (1, 1, 1, 1, 3, 3, 3, 3, 4)$

2.3 The Characterization of Erdős and Gallai

In order to identify degree sequences as not realizable, we can use the characterization of Erdős and Gallai [EG60]. Given a descending sorted degree sequence D, the outcome of the characterization is true if and only if the degree sequence is realizable. The characterization is defined as follows for a degree sequence $D = (d_1, \ldots, d_n)$ where $d_i \geq d_j$ for all $1 \leq i \leq j \leq n$ and $\sum_{i=1}^{n} d_i$ is even.

$$\forall r \in \{1, \ldots, n\} : \sum_{i=1}^{r} d_r \leq r(r-1) + \sum_{i=r+1}^{n} \min(d_i, r)$$

The characterization determines whether a degree sequence D is realizable by checking how many edges need to be added to the first r vertices on the left side of the inequality. The sum on the right side specifies the amount of edges that can be added among the first r vertices and between the first r vertices and the remaining $n - r$ vertices. If for each r the value on the left side is at most the sum on the right side, then D is realizable as enough edges can be added between the first r vertices and the least $n - r$ vertices, such that the degree of of each vertex $v_i = d_i$ for all $1 \leq i \leq r$.
We can use the characterization in order to verify the realizability of a solution degree sequence D' in two ways. First, we can use D' as input in order to identify solutions which cannot be realized even if the resulting graph was not restricted to be a supergraph of the input graph. For instance, the solution sequence for graph G_2 illustrated in Figure 4 is not realizable. Furthermore, we can use the difference sequence $D' - D$ as input where D is the initial degree sequence. In this way, solutions which cannot be realized as a supergraph G' of the input graph $G = (V, E)$ can be recognized. As we do not remove edges from the input graph, $D' - D$ is the degree sequence of vertex degrees of a graph $G'' = (V, E'')$ such that the degree sequence of $G' = (V, E \cup E'')$ is D'. If the outcome of the characterization using the difference sequence as input is not true, then there is no such graph G'' and we cannot realize the solution degree sequence within the input graph G. Otherwise, D' *might* be realizable within G. As existing edges within G might prevent the realization and the characterization does not consider the graph structure, realization can fail and the characterization might give a false positive. Figure 5 demonstrates an instance where the outcome of the characterization is true for the 2-anonymous degree sequence $D'_3 = (1, 1, 1, 1, 3, 3, 4, 4, 4)$ and the difference sequence $D'_3 - D_3$. However, D'_3 cannot be realized as a supergraph of G_3: We need to add an edge between two degree-three-vertices in order to create two degree-four-vertices, but each pair of degree-three-vertices is already adjacent.

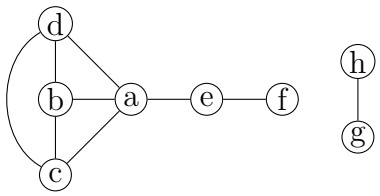


Figure 6: Graph $G_4 = (V_4, E_4)$, $D_4 = (1, 1, 1, 2, 3, 3, 3, 4)$

In order to improve the verification of realizability, we enhance the characterization of Erdős and Gallai such that the sum of the right side is reduced when we cannot add edges due to already existing edges. Therefore, we additionally store the initial degree d_i and the solution degree d'_i of each of the difference sequence's entries $d'_i - d_i$. Algorithm 1 illustrates our modification of the characterization. Among the first r entries, we check all mappings between these entries and the vertices in G with the vertex degree of the entries' initial degree and seek for a mapping, such that the sum of adjacencies among these vertices is minimal (line 7-16). For each edge in this mapping we subtract two from $r(r-1)$ on the right side of the inequality as we cannot add another edge in order to raise the vertex degree of the endpoints of this edge (line 13-14). The computation of the second sum (line 18-31) is as follows: For each of the least $n - r$ entries' initial degree, we iterate over the set of vertices whose degree is equal to the entry's initial degree (line 22). For each such vertex v_j, we seek for a mapping between the first r initial degrees $(d_1, \ldots, d_r)$ of the degree sequence and the vertices with this degree $\{v_1, \ldots, v_r\}$ (line 25-26), such that v_j is adjacent to as less vertices as possible from this set. If this sum of non-adjacent vertices is smaller than the minimum of d_j and r, we subtract $\min(\min(d_j, r) - \texttt{subsum}, 0)$ from the sum. However, we need to restrict this procedure: If an entry's initial degree is equal to another entry's solution degree, then we cannot reduce the regarding summand (line 20). Graph G_4 illustrated in Figure 6 is such an instance. We can realize $D'_4 = \{2, 2, 2, 3, 3, 4, 4, 4\}$ if we add the edges $\{\{\mathrm{d}, \mathrm{e}\}, \{\mathrm{f}, \mathrm{h}\}, \{\mathrm{c}, \mathrm{g}\}\}$. However, the advanced characterization fails without the restriction: Leaving out the vertices which remain unchanged, the difference sequence is $\{1, 1, 1, 1, 1, 1\}$, the corresponding initial degrees are $\{3, 3, 2, 1, 1, 1\}$ and the corresponding solution degrees are $\{4, 4, 3, 2, 2, 2\}$. However, for $r = 2$, the check would find out that all degree-3-vertices are adjacent, and subtract 2 from $r(r-1)$ on the right side. This would be wrong: When we first add the edge $\{\mathrm{e}, \mathrm{f}\}$, the degree of vertex e becomes 3. The set of vertex-3-degrees includes vertex e then, which is not adjacent to the other vertices with degree 3. This means that we cannot subtract 2 if the initial degree of an entry equals a solution degree of any another entry. The following pseudo code shows the functionality of the advanced Erdős Gallai characterization.

Algorithm 1 Pseudocode of Advanced Erdős and Gallai Characterization.

```
1: procedure ErdosGallaiAdvanced(G = (V, E), D, D')
2:     D^Δ ← descending sorted difference sequence D'_G − D_G
3:     r ← 1
4:     while r < n do
5:         leftsum ← Σ_{i=1}^{r} d_i^Δ
6:         rightmul ← r(r − 1)
7:         for all i ∈ {1, ... r} do
8:             if d_i ∉ D' then
9:                 minsub ← 0
10:                for all v ∈ V such that deg(v) = d_i do
11:                    sub ← 0
12:                    for j ∈ {i + 1, ... r} do
13:                        if ∄u ∈ V such that deg(u) = d_j and u, v not adjacent then
14:                            sub ← sub + 2
15:                    minsub ← min(minsub, sub)
16:                rightmul ← rightmul − minsub
17:        rightsum ← 0
18:        for all i ∈ {r + 1, ... n} do
19:            rightsum ← rightsum + min(r, d_i^Δ)
20:            if d_i ∉ D' then
21:                maxNonAdjacent ← 0
22:                for all v ∈ V such that deg(v) = d_i do
23:                    nonAdjacent ← 0
24:                    for all j ∈ {1, ... r} do
25:                        if ∃u ∈ V such that deg(u) = d_j and u, v not adjacent then
26:                            nonAdjacent ← nonAdjacent + 2
27:                    maxNonAdjacent ← min(maxNonAdjacent, nonAdjacent)
28:                if r ≤ d_i^Δ and maxNonAdjacent < r then
29:                    rightsum ← rightsum − (r − maxNonAdjacent)
30:                else if r > d_i^Δ and maxNonAdjacent < d_i^Δ then
31:                    rightsum ← rightsum − (d_i^Δ − maxNonAdjacent)
32:        if leftsum < rightsum then
33:            return not realizable
34:    return realizable
```

2.4 Limitations of Degree Sequence Anonymization Algorithms

Within this section, we show the limitations of using the presented degree sequence anonymization heuristics in Phase 1 of the graph anonymization process. First of all, we can use Algorithm 1 or Algorithm 2 in order to compute k-anonymous solution degree sequences with minimal cost and test these sequences for realizability with the advanced characterization of Erdős and Gallai. In order to ensure that the input graph cannot be turned into a k-anonymous graph with cost of s meaning that it is correct to con-

tinue the degree anonymization process with larger cost, it is necessary to verify that each possible solution degree sequence with $\texttt{distance}(D', D) = s$ cannot be realized using the characterization of Erdős and Gallai. As the backtracking algorithm used to compute the solution degree sequence based on the dynamic programming table computes exponentially many of these degree sequences with cost of s, we were not able to perform these checks in polynomial-time. Because our approach is to k-anonymize a graph in polynomial-time, we had to abort the anonymization process after decent time as increasing the cost might give non-optimal results. Within the algorithm engineering process, we encounter two strategies to solve this problem. The first approach is to decrease the size of the input degree sequence using data reduction rules, which is discussed in Section 3. The second approach is not to distinguish between the vertices of the same degree. This led to the concept of *degree block sequences*, where an entry stands for the amount of vertices of a specific degree. As the development of an algorithm for k-anonymizing such a degree block sequence goes beyond the scope of this bachelor thesis, this subject has been worked out by Hartung et al. [HHN14]. Hence, we use the algorithm for anonymizing degree block sequences as a black box and concentrate on the realization of k-anonymous degree block sequences in Section 4 and Section 5.

3 Data Reduction Rules

Using Algorithm 1 or 2, the amount of solution degree sequences which can be generated with minimal cost can be very large. We provide two Data Reduction Rules, which reduce the size of the input degree sequence such that the table used within the algorithm is reduced in size and the amount of solution degree sequences is reduced. Furthermore, decreasing the size of the degree sequence also improves the performance and reduces the memory consumption of Algorithm 1 and 2, as the running-time depends on the amount of entries within a degree sequence. Within this section, a *degree block* c_i denotes the amount of vertices within a graph with $\texttt{deg}(v) = i$.

Rule 1. If the size of a block c_i is at least $2s + k$, then remove all but $2s + k$ entries of this block.

Lemma 3. Rule 1 is correct.

Proof. Within computation of a k-anonymous degree sequence D', $2s$ determines the distance of an initial and a solution degree sequence. It is clear that at most $2s$ entries can be modified. If the degree of $p \leq 2s$ vertices with a specific degree is raised, then we only raise the least p entries of this degree. If there are more than $q > 2s$ entries of a specific value, then we will not modify the value of the first $q - 2s$ entries. As there need to remain at least k entries, we remove the first $b - k$ entries from the degree sequence. □

Rule 2. If Rule 1 was applied such that there is at least one block b_i with size $2s + k$, then remove all but $k + 2(k - 1)$ entries of each other block.

		s						
		100	200	500	1000	5000	10000	20000
	2	1.347	1.547	2.147	3.147	11.147	21.147	41.147
	5	3.051	3.251	3.851	.4851	12.851	22.851	42.851
	10	5.331	.5531	6.131	7.131	15.131	25.131	45.131
k	20	9.156	9.356	9.956	10.956	18.956	28.956	48.956
	50	18.538	18.738	19.338	20.338	28.338	38.338	58.338
	100	31.197	31.397	31.997	32.997	40.997	50.997	70.997
	200	38.872	52.569	53.169	54.169	62.169	72.169	92.169

Table 1: Size of the degree sequence of graph cit-Patents (3.774.768 vertices, $\Delta = 793$) after applying Data Reduction Rule 1 and 2.

Lemma 4. Rule 2 is correct.

Proof. If a block c has size $i < k$, then either all i vertices within this block need to be raised to the next block d which is not empty or $k - i$ vertices of the previous block b need to be raised to block c. In the latter case, the block b needs to contain at least $2k - 1$ vertices, such that after raising at most $k - 1$ vertices to block c there are still at least k vertices left in block b. As it might be necessary to raise the degree of at most $2s$ vertices in order to increase the amount affected vertices, there is one large block e with size $2s + k$ left. If more than $i > k$ vertices need to be raised for affecting vertices, then there are $2s \geq i$ vertices in the large block whose degree can be raised by one. If $i < k$ vertices need to be raised, then it might not be optimal to raise the degree of i vertices of block e: If the block following e is empty, then raising the degree of i vertices of block e forms a new block with $i < k$ degrees. As we ensure that the size of each large block is reduced to $k + 2(k - 1)$, then there are $k - 1$ vertices left in each such block which can be used to increase the amount of affected vertices. □

Using both Data Reduction Rule 1 and 2 decreased the size of power law distributed input degree sequence enormously. Table 1 shows the size of the reduced degree sequence of graph *cit-Patents*, which has 3.774.768 vertices, 16.518.947 edges and a maximum degree of 793. The downside of using these Data Reduction Rules for decreasing the size of the input degree sequences for Algorithm 1 or 2, we need to compute a reduced degree sequence for each increasing s, as the data reduction rules depend on the parameter s. The anonymization process then has to be restarted from scratch. Furthermore, if Algorithm 1 or 2 computes a k-anonymous solution degree sequence using data reduction rules and these degree sequences cannot be realized, then the degree anonymization process cannot be continued while increasing the cost s: If realization fails because of existing edges to degrees of the block with size $2s + k$, then selecting vertices of another degree within Data Reduction Rule 1 which form the large block might succeed. As neither Algorithm 1 nor Algorithm 2 compute realizable degree sequences for the graphs

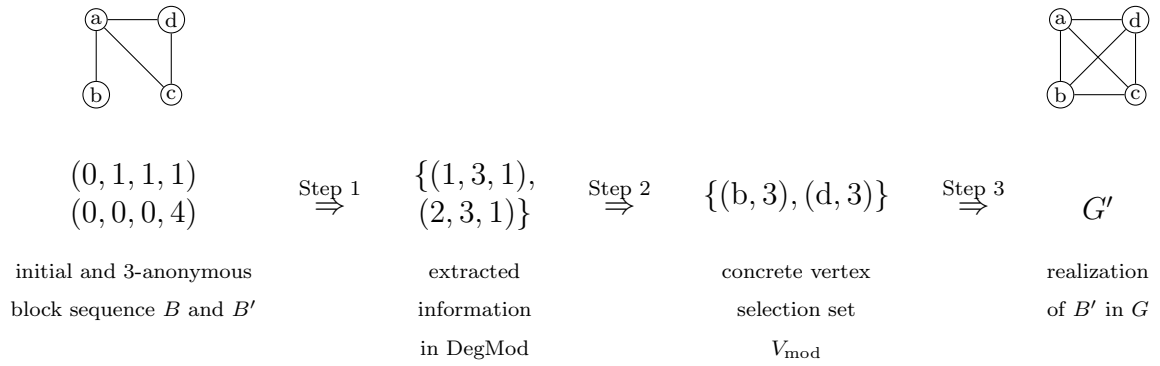


Figure 7: The three steps of degree block sequence realization.

in our test set, using the data reduction rules consequently did not improve the chances on realizability.

4 Concept and Realization of Degree Block Sequences

In this section, we introduce the concept of representing a degree sequence as a degree block sequence. Additionally, we introduce a strategy to extract information necessary to realize a k-anonymous block sequence within the input graph. The concept of degree block sequences has been used for degree sequence anonymization by Hartung et al. [HHN14]. This algorithm uses a slightly modified version of the Advanced Characterization of Erdős and Gallai in order to check the realizability of computed sequences. The main advantage of degree block sequences is that we the amount of computed solution degree block sequences can be reduced drastically compared to Algorithm 1 and 2 by using data reduction rules and lower bound heuristics. Furthermore, the size of a degree block sequence depends on the maximum degree of a graph instead of the number of its vertices. As many large power law distributed graphs have a rather low maximum degree, using degree block sequences is more efficient regarding to memory usage. However, additional steps are required when realizing a k-anonymous degree block sequence as a supergraph of the input graph. We split this realization process into three major steps. In Step 1, we determine how many vertices of a certain degree need to be modified by adding a certain amount of edges. This step is discussed in Section 4.1. In Step 2 discussed in Section 4.2, we heuristically select concrete vertices. This step can also be performed for k-anonymous degree sequences computed by Algorithm 1 or 2, if the mapping between the entries in the degree sequence and the vertices in the input graph has not been determined before the anonymization process. In Step 3 discussed in Section 5, we add edges between the selected vertices such that their degrees comply with the k-anonymous degree block sequence. Figure 7 illustrates an overview of realizing degree block sequences. As previously mentioned, we are using the dynamic programming algorithm from [HHN14] as a black box. Within this section, we refer to this algorithm as *Algorithm 3*.

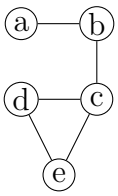


Figure 8: Graph $G_5 = (V_5, E_5)$, $k = 2$, $D_5 = (1, 2, 2, 2, 3)$

Notation. A block sequence is denoted as $B_G = (b_0, \ldots, b_\Delta)$. An entry $b_i \in B$ stores the amount of vertices with degree i. We can compute a block sequence from the input graph by counting the degrees of its vertices. Furthermore, a degree sequence can be transformed to a block sequence. For all $0 \leq i \leq \Delta$, an entry b_i is computed by counting the entries of a degree sequence whose value is i. A k-anonymous block sequence is denoted as $B'_G = (b'_0, \ldots, b'_\Delta)$. A block sequence B' is k-anonymous, if each $b'_i \in B'$ is either 0 or at least k. A k-anonymous block sequence B' *derives* from an initial block sequence B, if $\sum_{i=0}^{\Delta} b'_i - b_i = 0$, meaning that the amount of vertices represented by B and B' needs to be equal. Furthermore, for all $1 \leq r \leq \Delta$: $b'_r - b_r \leq \sum_{i=0}^{r-1} b_r - b'_r$.

4.1 Extracting Information from Initial and Solution Block Sequences

Once a k-anonymous block sequence B' is computed, we need to determine how many vertices of a certain degree within the input graph need to be modified, such that B' is the corresponding block sequence of the resulting k-anonymous graph. This information is stored as a sequence of triples, each of which contains the initial degree, the temporary solution degree and the amount of vertices whose degree is raised from this initial to solution degree. We denote this sequence as *DegMod*. This sequence is computed by comparing the initial degree block sequence B to the solution degree block sequence B' entry by entry. For each $0 \leq i \leq \Delta$, if $b_i > b'_i$, then we know that the degree of $b_i - b'_i$ vertices with degree i needs to be increased. The temporary solution degree for these vertices is the first index $j > i$ such that $b'_j > 0$. Hence, we add $(i, j, b_i - b'_i)$ to DegMod and set b_j to $b_j + (b_i - b'_i)$.

Example 4.1. The initial block sequence of Graph G_5 shown in Figure 8 is $B = (0, 1, 3, 1)$, meaning that there is one vertex of degree 1, three vertices of degree 2 and one vertex of degree 3 in the input graph. The 2-anonymous block sequence computed by Algorithm 3 is $B' = (0, 0, 3, 2)$ meaning that one edge needs to be added between the vertex with degree 1 and a vertex with degree 2. Hence, the computed sequence is DegMod $= ((1, 2, 1), (2, 3, 1))$ meaning that the degree of a vertex with degree 1 needs to be raised from 1 to 2 and the degree of a vertex with degree 2 needs to be raised from 2 to 3. The sequence DegMod forms the basis of Step 2, where concrete vertices are selected for each entry.

Example 4.2. Graph G_6 illustrated in Figure 9 has the initial block sequence $B = (2, 2, 0, 4))$. For $k = 3$, the solution block sequence is $B' = (0, 3, 0, 5)$. The degree raise

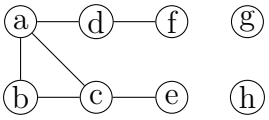


Figure 9: Graph $G_6 = (V_6, E_6)$, $B_6 = (2, 2, 0, 4)$, $k = 3$

sequence DegMod must necessarily contain the entries $(0, 1, 2)$ and $(1, 3, 1)$, meaning that the degree of one vertex with degree 1 needs to raised to 3 and both vertices with degree 0 need to be added one edge-vertex-adjacency to. This is realizable if we add the edges $\{6, 7\}$ and $\{6, 8\}$. If the degree raise sequence contained $(0, 1, 1)$ and $(0, 3, 1)$ instead, the maximum raise of 3 was larger than the amount of affected vertices which is 2 and there is no vertex assignment such that this sequence is realizable.

4.2 Computing Mappings between Vertices and DegMod

After computing the degree raise sequence DegMod which stores information about how many vertices a of a certain degree p need to be raised to a specific target degree $q > p$, we need to compute a mapping between the vertices of the input graph and the entries in DegMod. The entries $(p_1, q_1, a_1), \ldots, (p_m, q_m, a_m)$ are sorted in ascending order, such that $p_i < p_j$ for all $i < j$. The positive integer m denotes the number of entries in DegMod. The sequence is processed in the order specified. For each entry (p_i, q_i, a_i) we need to select a_i vertices of degree p_i, whose degree needs to be raised to q_i. These vertices are stored alongside the target degree in a set V_{mod}. Note that for finding a bijective mapping between the entries of a k-anonymous degree sequence D' and the vertices of the input graph, DegMod can simply be computed by comparing the initial degree sequence D to D': For each $i \in \{1, \ldots, n\}$, if $d'_i > d_i$ then a triple $(d_i, d'_i, 1)$ is added to DegMod.

The set V_{mod} is computed as follows. If the amount of vertices of degree p_i equals a_i, then all of these vertices alongside their initial degree p_i and the target degree q_i are added to V_{mod}. Otherwise, if $b_{p_i} < a_i$, then there are not enough vertices which initially have degree p_i. However, then at least $a_i - b_{p_i}$ vertices have been selected to be raised to p_i in a previous step. Hence, we need to select at least $a_i - b_{p_i}$ of these vertices and raise them to q_i instead of p_i. Otherwise, we need to raise a_i vertices of degree p_i to q_i and there are enough vertices in the input graph with degree p_i, meaning that we need to select a_i of these vertices. We introduce three strategies to arrange this selection. *Selection Strategy 1* checks for each vertex of degree p_i, to how many vertices in V_{mod} this vertex is adjacent. The vertices with the fewest neighbors in this set are selected, as adjacencies between the vertices in V_{mod} usually lower the chances on realizability. *Selection Strategy 2* sums for each vertex of degree p_i the degrees of all its neighbors. If a vertex is mainly connected to low-degree vertices, the sum is rather low and these vertices are preferred then. *Selection Strategy 3* is a randomized strategy, which selects any a_i vertices of degree p_i.

As the set V_{mod} is empty when we start the selection procedure, Selection Strategy 1 cannot distinguish between the vertices available in the beginning of the selection

process. In order to improve the computed solution after all entries of DegMod have been processed, we reiterate the computed set V_{mod}. For each vertex in V_{mod}, we count the number of neighbors within this set. If this number is larger than 0, we try to swap the vertex for another vertex of the same initial degree which is not yet in V_{mod} and which has less neighbors within this set. For some instances, we achieved improvements regarding to realizability if we repeat this step until no vertices can be swapped any more. This modification of Strategy 1 is referred to as *Strategy 1 Refined*.

5 Greedy Graph Constructor Algorithm

This section covers the realization of a k-anonymous degree sequence respectively block sequence, which is part of Phase 2 of k-anonymizing a graph. The input of the graph constructor algorithm is a set of vertices V_{mod} and the target degree of each vertex in this set. Before performing the graph construction process, V_{mod} needs to be computed according to Section 4.2. The procedure of realizing degree (block) sequences described within this section is also referred to as *graph construction*. We introduce several strategies which differ in the order edges are added to the vertices within the set V_{mod} and proof that the graph construction process always succeeds if the amount of vertices within this set is larger than $2 \cdot \Delta^4 + \Delta^2 + 1$.
The basic idea of our graph construction procedure is to add edges to the input graph $G = (V, E)$ until the degree of each vertex $v \in V_{\text{mod}}$ has reached its target degree. We denote the target degree of a vertex $v_i \in V_{\text{mod}}$ as t_i. The procedure of graph construction is done in two steps. In Step 1, for each pair of non-adjacent vertices $u, v \in V_{\text{mod}}$, we add an edge between u and v if the vertex degree of both u and v (still) needs to be raised. We implemented three strategies, which differ in the order these vertices are selected. Step 2 is performed if there are still vertices left, whose degree could not be increased to its desired degree in Step 1. This can occur if some vertices within V_{mod} are already adjacent before Step 1 or if an edge has previously been added between these vertices. In order to realize the k-anonymous (block) sequence, we try to remove an added edge $\{s, t\}$ in order to add two other edges $\{s, u\}$ and $\{t, v\}$, where $u, v \in V_{\text{mod}}$ and the degree of both u and v needs to be increased.

Details. Algorithm 2 illustrates the pseudo code of Greedy Graph Constructor. In Step 1, the algorithm checks for each vertex $v_i \in V_{\text{mod}}$ whether it has already reached its target degree t_i. If it has not, and if there are at least t_i other vertices $v_j \in V_{\text{mod}}$ such that $\texttt{deg}(v_j) \neq t_j$ and v_i is not adjacent to v_j, we select t_i of these vertices and add an edge $\{v_i, v_j\}$ for each selected vertex v_j (line 5-12). Otherwise, if there are less than t_i vertices within V_{mod} which are not adjacent to v_i and which have not yet reached their target degree, we add an edge between v_i and each of these vertices. As we are not able to raise the degree of v_i to its target degree in Step 1, we try to correctly solve the graph construction in Step 2.
The order in which the selected vertices are processed has a strong effect on realizability. We implemented three different strategies which differ in the order in which the vertices

within V_{mod} are processed. *Processing Strategy 1* has turned out to be the most effective strategy regarding to realizability. It first processes the vertices where the difference between the target degree and the initial degree is largest. Additionally, we implemented an enhanced version of this strategy which changes the processing order after each added edge. As adding an edge decreases the difference between initial and solution degree of two vertices, the difference of other vertices might become larger than the difference of the most recently processed vertices. The downside of this modification referred to as *Processing Strategy 1 Refined* is a slight increase of running-time, however this gave slightly better results for some instances regarding to realizability.
Processing Strategy 2 first processes the vertices whose amount of adjacent vertices within V_{mod} is largest. *Processing Strategy 3* is a randomized strategy, which processes the vertices within V_{mod} in random order. If Strategy 3 is used, we recommend performing the graph construction process several times if it was not successful.
Our experiments revealed that Processing Strategy 2 and 3 did not achieve better results than Processing Strategy 1 for any of the tested instances. Furthermore, the running-time of Processing Strategy 3 is much larger than the one of Processing Strategy 1. Hence, we omit presenting the experimental results for these strategies.

Step 2 of graph construction process is performed if there is at least one vertex $v_i \in V_{\text{mod}}$ such that $\texttt{deg}(v_i) \neq t_i$. We distinguish between two cases. In case one, there are at least two vertices $v_i, v_j \in V_{\text{mod}}$, such that v_i and v_j have not yet reached their target degree (line 13-23). In order to increase $\texttt{deg}(v_i)$ and $\texttt{deg}(v_j)$, the algorithm tries to remove a previously added edge $\{s, t\}$ and adds two edges $\{s, v_i\}$ and $\{t, v_j\}$ instead. This step is repeated as long as there exist such vertices v_i, v_j, s, t. In case two, there is exactly one vertex v_i where $\texttt{deg}(v_i) \neq t_i$ (line 24-31). The difference between $\texttt{deg}(v_i)$ and the target degree must be even, as $\sum_{i=1}^{n} t_i - \texttt{deg}(v_i)$ is even before any edges are added due to the degree anonymization algorithm. Furthermore, each added edge reduces this sum by exactly 2. Similarly to case one, we try to remove a previously added edge $\{s, t\}$ in order to add $\{s, v_i\}$ and $\{t, v_i\}$. This procedure increases the degree of v_i by 2 and we repeat this step until $\texttt{deg}(v_i) = t_i$ and there is at least one such edge $\{s, t\}$. If $t_i = \texttt{deg}(v_i)$ for all $v_i \in V_{\text{mod}}$ then graph construction was successful and the resulting graph is k-anonymous (line 32). The set of added edges is the solution set. Otherwise, there is still at least one vertex left which has not reached its target degree and Greedy Graph Constructor Algorithm is not able to realize the k-anonymous degree (block) sequence with the used strategies for vertex selection and vertex processing in Step 1. We need to remove all added edges and repeat the graph construction process with another vertex selection strategy described in Section 4.2. Furthermore, we can use another vertex processing strategy in Step 1 of the graph construction process. However, the k-anonymous sequence might not be realizable at all within the input graph. In this case, graph construction will fail with each combination of vertex selection and processing strategies. Our experiments revealed that in this case a slight modification of the parameter k can result in computation of realizable degree block sequences (see Section 6 for details).

Algorithm 2 Pseudocode of Greedy Graph Constructor.

procedure GREEDY GRAPH CONSTRUCTOR($G = (V, E)$, V_{mod})
  $V_{\text{todo}} \leftarrow V_{\text{mod}}$
  $V_{\text{touched}} \leftarrow \emptyset$
  $S \leftarrow \emptyset$
  while $\exists u, v \in V_{\text{todo}}$ such that $\{u, v\} \notin E \cup S$ **do** // Step 1: adding edges
    $S \leftarrow S \cup \{\{u, v\}\}$
    $V_{\text{touched}} \leftarrow V_{\text{touched}} \cup \{u, v\}$
    $t_u \leftarrow t_u - 1$, $t_v \leftarrow t_v - 1$
    if $t_u = 0$ **then**
      $V_{\text{todo}} \leftarrow V_{\text{todo}} \setminus u$
    if $t_v = 0$ **then**
      $V_{\text{todo}} \leftarrow V_{\text{todo}} \setminus v$
  while $\exists u, v \in V_{\text{todo}}$ and $\exists s, t \in V_{\text{touched}}$ such that $\{s, t\} \in S$ and $\{\{s, u\}, \{t, v\}\} \notin E$ **do**
      // Step 2: adding edges by reconnecting if at least two vertices left in V_{todo}
    $S \leftarrow S \setminus \{\{s, t\}\}$
    $S \leftarrow S \cup \{\{s, u\}\}$
    $S \leftarrow S \cup \{\{t, v\}\}$
    $V_{\text{touched}} \leftarrow V_{\text{touched}} \cup \{u, v\}$
    $t_u \leftarrow t_u - 1$, $t_v \leftarrow t_v - 1$
    if $t_u = 0$ **then**
      $V_{\text{todo}} \leftarrow V_{\text{todo}} \setminus u$
    if $t_v = 0$ **then**
      $V_{\text{todo}} \leftarrow V_{\text{todo}} \setminus v$
  while $\exists u \in V_{\text{todo}}$ and $\exists s, t \in V_{\text{touched}}$ such that $\{s, t\} \in S$ and $\{\{s, u\}, \{t, u\}\} \notin E$ **do**
      // Step 2: adding edges by reconnecting if exactly one vertex left in V_{todo}
    $S \leftarrow S \setminus \{\{s, t\}\}$
    $S \leftarrow S \cup \{\{s, u\}\}$
    $S \leftarrow S \cup \{\{t, u\}\}$
    $t_u \leftarrow t_u - 2$
    if $t_u = 0$ **then**
      $V_{\text{todo}} \leftarrow V_{\text{todo}} \setminus u$
  if $V_{\text{todo}} = \emptyset$ **then**
    $E \leftarrow E \cup S$
    return S
  else
    return Realization not successful

Lemma 5. Greedy Graph Constructor runs in $O(n^4)$ time.

Proof. Adding edges in Step 1 is performed in $O(\Delta^2 \cdot n)$ time: There are $O(n)$ vertices in V_{mod} as the graph consists of n vertices. As the maximum degree in a k-anonymous degree sequence is at most Δ^2 [HHN14], we add up to $O(\Delta^2)$ edges to each vertex in V_{mod}. Hence, Step 1 runs in $O(n\Delta)$ time.
Reconnecting edges in Step 2 can be done in $O(n^4)$ time. There are $O(n)$ vertices

in V_{mod} which have not yet reached their target degree after Step 1. Hence, there are $O(n * (n-1)/2)$ pairs of such vertices. Furthermore, $O(n \cdot (n-1)/2)$ edges have been added previously. For each pair of vertices v_i, v_j within V_{mod} which have not yet reached the target degree, the algorithm tries to remove an added edge $\{s,t\}$ in order to perform the reconnection. In order to find such an edge $\{s,t\}$, we need to process at most all previously added edges. In summary, one iteration is performed in $O(n * (n-1)/2)^2 = O(n^4)$ time. □

Lemma 6. Greedy Graph Constructor successfully constructs a k-anonymous graph G' if V_{mod} contains at least $2 \cdot \Delta^4 + \Delta^2 + 1$ vertices.

Proof. We denote the set of vertices, whose degree has already been raised by adding at least one edge, by V_{touched}. The set of vertices, whose degree has not reached its target degree yet, is denoted as V_{todo}. Initially, $V_{\text{todo}} = V_{\text{mod}}$ and V_{touched} is empty.
The target degree of any vertex in V_{mod} is at most Δ^2, as the maximum degree within a k-anonymous degree (block) sequence computed by either Algorithm 1 or 2 or the algorithm of Hartung et al. is Δ^2 [HHN14]. Hence, the vertex degree of any vertex in V_{todo} is at most $\Delta^2 - 1$. If V_{todo} contains at least Δ^2 vertices, at least two vertices v_i and v_j in this set are not adjacent. In this case, the algorithm adds an edge between such a pair of vertices in Step 1. Both vertices then are contained in V_{touched}.

If Step 1 cannot be performed and V_{todo} contains at least two vertices, the algorithm is able to remove a previously added edge in order to add two edges in Step 2 if the size of V_{touched} is at least $(\Delta^2 - 2) \cdot (\Delta^2 - 1) + 2$: For two vertices $u_1, u_2 \in V_{\text{todo}}$, there are two vertices $s, t \in V_{\text{touched}}$, such that neither s nor t are adjacent to both u_1 and u_2. The vertices u_1 and u_2 are adjacent to at most $\Delta^2 - 2$ vertices in $V_{touched}$ each, as the maximum degree of vertices in V_{todo} is $\Delta^2 - 1$ and u_1 and u_2 are already adjacent. In the worst case, u_1 and u_2 are adjacent to the same $\Delta^2 - 2$ vertices in V_{touched}, and each such vertex is adjacent to $\Delta^2 - 1$ other vertices in V_{touched}. If V_{touched} contains at least $(\Delta^2 - 2) \cdot (\Delta^2 - 1) + 2$ vertices, there is always at least one pair of vertices s, t which is adjacent and neither s nor t is adjacent to both u_1 and u_2.
Otherwise, if V_{todo} contains exactly one vertex u, we need to remove a previously added edge $\{s,t\}$ in order to add the edges $\{s,u\}$ and $\{t,u\}$. To make sure that this procedure can be performed, neither s nor t may be adjacent to u. If V_{touched} contains at least $(\Delta^2 - 1)^2 + 1$ vertices, then there are at least two adjacent vertices s, t which are not adjacent to u, as u is adjacent to at most $\Delta^2 - 1$ vertices in V_{touched} and each of these vertices is adjacent to at most $\Delta^2 - 1$ other vertices in V_{touched}. If there are two more vertices in V_{touched}, then these vertices have to be adjacent to each other but not to u. Hence, reconnection can be performed.
In summary, Greedy Graph Constructor Algorithm successfully solves an instance if V_{mod} initially contains at least $\Delta^2 + (\Delta^2 - 1)^2 + 1 = \Delta^4 - \Delta^2 + 2$ vertices. □

6 Implementation and Experimental Results

Within this section, we present the implementation and the experimental results for degree block anonymization and graph construction. As previously mentioned, we used the degree block sequence anonymization implementation of [HHN14] as a black box. Hence, we do not provide further details of their implementation within this work. Furthermore, we don't consider experimental results for Algorithm 1 and 2 within this section, as we were not able to realize k-anonymous solution sequence computed by these algorithms. The focus of our experiments was to rate the various strategies for graph realization.
All of our software was written in Java 7 and runs under the OpenJDK runtime environment 1.7.0_25. The experimental tests were processed on Intel Xeon E5-1620 3.6GHz machines with 64GB memory under the Debian GNU/Linux 6.0 operating system. Our test suite consists of five *Co-author and Citation Networks* graphs from the 10^{th} DIMACS challenge, eight network graphs from the *Stanford Large Network Dataset Collection* and eleven coauthor networks derived from the DBLP dataset, which was generated on February 2012 following the documentation from http://dblp.uni-trier.de/xml/. Table 2 provides some statistics of the used graphs. The integer n denotes the amount of vertices within a graph and m denotes the amount of edges. The maximum degree is denoted by Δ.

6.1 Implementation

Our implementation was written in Java 7 using the OpenJDK environment. The program reads various command line arguments such as input graph file, k, logging level, output file, vertex selection and processing strategy and a flag which enables data reduction rules for the degree block sequence anonymization. Input graphs are read as text files and stored as an array of adjacency set: For each vertex, there is a Hash Set which represents the set of adjacent vertices. As we use 32-bit integers to reference a vertex, we decided to use the Trove library 3.03 which offers Hash Sets of native data types instead of storing Integer Objects, which use at least 8 Bytes of memory each. Thus, we were able to store even very large social network graphs with decent usage of memory. Furthermore, querying the graph data structure in order to add and remove edges or to determine the degree of a vertex can be done in $O(1)$. The dynamic programming Algorithm 1 and 2 presented in Section 2.1 were implemented using tables of *BitSets*. As the tables' size depends on the amount of vertices of the input graph and each entry stores a Boolean value, using BitSets turned out to be quite efficient with regard to running-time and memory usage. However, as none of the resulting k-anonymous degree sequences of our test graphs could be realized, we omit further results within this section. The dynamic programming algorithm used to compute k-anonymous block sequences has been programmed with Java 7, but was used as a black box. Hence, we do not offer further implementation details of the implementation within this work. Other entities such as degree raise sequences and sets of selected vertices are stored as ArrayLists of integers or classes which store several integers. The graph construction implementation

Table 2: Statistics about the used graphs within our experiments.

Social network graphs				
	Graph	n	m	Δ
DIMACS graphs	coAuthorsCiteseer	227.320	814.134	1.372
	coAuthorsDBLP	299.067	977.676	336
	coPapersCiteseer	434.102	16.036.720	1.188
Stanford social networks	cit-HepPh	34.546	420.877	846
	cit-Patents	3.774.768	16.518.947	793
	com-Amazon	334.863	925.872	549
	com-DBLP	317.080	1.049.866	343
	facebook-combined	4.039	88.234	1.045
coAuthor networks	graph_authorFilter_thres_02	1.146	2.652	99
	graph_authorFilter_thres_03	699	1.224	71
	graphConference	5.599	8.492	53
	graph_conferenceFilter_thres_01	11.133	27.255	149
	graph_conferenceFilter_thres_02	2.917	4.066	65
	graph_conferenceFilter_thres_03	1.248	1.348	43
	graph_thres_01	715.633	2.511.988	804
	graph_thres_02	282.831	640.697	201
	graph_thres_03	167.006	293.796	123
	graph_thres_04	112.949	168.524	88

uses HashMaps and HashSets, which store information about added edges. Hence, performing a lookup whether an edge has been added during graph construction can be done in $O(1)$.

6.2 Results

Testing method. Our experimental tests were done as follows. For each of the tested input graphs and each $k \in \{2, 3, 4, 5, 10, 20, 50, 100, 200\}$, we first computed a set of k-anonymous degree block sequences using the degree block anonymization algorithm of Hartung et al. [HHN14]. If the anonymization did not finish within 20 minutes, we aborted the anonymization process and continued with another graph or the next value for k. The degree block anonymization implementation is limited to compute at most 500 k-anonymous degree block sequences. Moreover, the anonymization can be started using internal data reduction rules, which improves running-time and reduces the amount of computed solutions for some instances. We tried to realize each solution as a supergraph of the input graph with Selection Strategy 1 and Selection Strategy 1 Refined (see Section 4.2 for details) together with each Processing Strategy introduced in Section 4. The randomized Selection Strategy 3 was run at most twenty times for each instance, which turned out to be a good trade-off between running-time and effectiveness. If the realization process did not finish after at most 30 minutes, we aborted the process.

Table 3: Results for degree block anonymization. For each graph and each k, the first entry of each column is the amount of computed k-anonymous degree block sequences using internal data reduction rules within 20 minutes. The second entry is the amount of computed k-anonymous degree block sequences without using internal data reduction rules using internal data reduction rules. If an entry is 0, then the anonymization process has been aborted after 20 minutes.

Graph	Parameter k								
	2	3	4	5	10	20	50	100	200
coAuthorsCiteseer	118 | 500	500 | 500	500 | 500	0 | 0	0 | 0	500 | 0	0 | 0	0 | 0	0 | 0
coAuthorsDBLP	3 | 500	500 | 500	500 | 500	500 | 500	500 | 500	500 | 500	0 | 0	0 | 0	1 | 1
coPapersCiteseer	500 | 0	1 | 1	1 | 1	5 | 16	1 | 64	1 | 1	1 | 1	1 | 1	2 | 0
cit-HepPh	500 | 0	0 | 0	0 | 0	0 | 0	0 | 0	0 | 0	0 | 0	0 | 0	0 | 0
cit-Patents	500 | 500	0 | 0	10 | 120	1 | 1	2 | 2	0 | 0	0 | 0	0 | 0	0 | 0
com-Amazon	12 | 500	500 | 0	0 | 0	0 | 0	0 | 0	0 | 0	500 | 0	500 | 0	500 | 500
facebook-combined	118 | 500	9 | 0	0 | 0	0 | 0	0 | 0	0 | 0	0 | 0	0 | 0	0 | 0
graph_authorFilter_thres_02	24 | 48	2 | 10	30 | 210	58 | 58	65 | 65	1 | 96	0 | 0	1 | 1	1 | 1
graph_authorFilter_thres_03	2 | 12	19 | 57	60 | 60	74 | 74	78 | 78	11 | 11	3 | 3	1 | 1	1 | 1
graphConference	1 | 8	5 | 25	333 | 333	121 | 242	500 | 500	500 | 500	500 | 500	1 | 1	1 | 1
graph_conferenceFilter_thres_01	2 | 20	17 | 59	2 | 20	246 | 246	500 | 500	500 | 500	500 | 500	1 | 91	1 | 1
graph_conferenceFilter_thres_02	2 | 2	2 | 2	1 | 2	5 | 10	18 | 18	13 | 13	1 | 1	1 | 1	1 | 1
graph_conferenceFilter_thres_03	1 | 1	2 | 2	2 | 2	7 | 7	5 | 5	4 | 4	1 | 1	1 | 1	1 | 1
graph_thres_01	0 | 0	0 | 0	0 | 0	0 | 0	0 | 0	0 | 0	0 | 0	500 | 0	1 | 0
graph_thres_02	1 | 1	1 | 1	15 | 1	1 | 1	2 | 2	500 | 500	1 | 1	1 | 1	1 | 1
graph_thres_03	5 | 30	40 | 40	76 | 151	312 | 500	500 | 500	500 | 500	1 | 1	1 | 1	1 | 1
graph_thres_04	11 | 44	79 | 79	102 | 102	1 | 1	49 | 49	1 | 1	1 | 1	1 | 1	1 | 1

Results. Depending on the parameter k, we were able to compute 200-anonymous degree block sequences for most of the graphs in our test set. We observed that the number of vertices and the maximum degree have the largest impact on running-time of the algorithm. Hence, most graphs with a large maximal degree mostly could not be k-anonymized within 20 minutes for a $k \geq 5$. Furthermore, we observed that if anonymization takes much time or the resulting degree block sequences are not realizable for an instance (G, k), then slightly modifying the parameter k often has a great impact on the following realization procedure.

For the degree sequence realization procedure, selecting vertices randomly (Selection Strategy 3) was the best strategy for most instances of our test set. Even with 20 randomized passes, this strategy was superior to each other selection strategy in average and could realize up to 37.0% of the instance in our test set. However, the average time for realizing a degree block sequence (319.5s) is about 50% larger than the one of Selection Strategy 1. Selecting vertices which have the fewest common neighbors within the selection (Selection Strategy 1) turned out to be the fastest realization strategy while being almost as effective as the randomized strategy regarding to realizability. Improving this selection as long as no vertices can be swapped for other ones with fewer neighbors within the selection (Selection Strategy 1 Refined) even increased the amount of successful realization for some instances while increasing the running-time by about 5%. Selecting vertices whose neighbors have the lowest degree (Selection Strategy 2) turned out to be the most ineffective strategy, while average running-time is similar to the one of Selection Strategy 1. In theory, Selection Strategy 1 should be superior to any

Table 4: Results for degree block anonymization using internal data reduction rules. The entries indicate how many k-anonymous degree block sequences were computed within 20 minutes. If an entry is 0, then the anonymization process has been aborted after 20 minutes.

Selection Strategy	Processing Strategy	Realizable Instances	Average Running Time (s)
1	1	33.0%	57.6
1	1 ref.	34.6%	203
2	1	32.4%	55.1
2	1 ref.	34.0%	208
3	1	36.5%	189
3	1 ref.	37.0%	319
1 ref.	1	33.1%	59.6
1 ref.	1 ref.	34.8%	203.6

other strategy as existing edges between the selected vertices in the input graph lower the chances on realizability. If there is no such edge at all, then a selection is perfect and realizable due to the usage of the characterization of Erdős and Gallai within the anonymization process. However, for most instances we were not able to compute a vertex selection set where no vertices are adjacent. Hence, the strategy is a heuristic in this case and there might be better selections.

The strategy which determines the order of adding edges between the selected vertices (Processing Strategy) has a very large impact on realizability. As mentioned in Section 5, firstly processing vertices whose difference between initial and target degree is the largest (Processing Strategy 1) turned out to be the most effective strategy regarding to realizability. Refining this strategy by recalculating the order after each added edge (Processing Strategy 1 Refined) even improves the chances on realizability for some instances. In average, using Processing Strategy 1 Refined increased the amount of realized degree sequences by about 5% but also increased the running-time by 70% to 350%, depending on the used selection strategy. In contrast, firstly processing the vertices which have the most neighbors within the set of selected vertices (Processing Strategy 2) turned out to be very ineffective and realization failed for most instances. Randomly processing vertices (Processing Strategy 3) turned out to be even more ineffective. This seems logical as due to the power law distribution of the tested graphs and their degree sequences, there are very few vertices whose degree needs to raised by a large amount and there are quite many vertices which must be raised by a rather small amount. If last-named vertices are processed first, then there are not enough vertices left which the high-degree vertices can be made adjacent to and realization fails. Moreover, for power law distributed graphs, the amount of affected vertices within a k-anonymous degree (block) sequence was almost equal to the maximum raise. Hence, vertices which need many edges to be added consequently need to be processed at first. We omit presenting the experimental results of Processing Strategy 2 and 3, as Processing Strategy 1 was superior regarding to effectiveness and running-time for any tested instance.

6.3 Conclusion and Outlook

Using the degree block anonymization algorithm of Hartung et al. [HHN14] and the heuristic graph construction algorithms introduced within this paper, we were able to make many social network graphs k-degree anonymous. Due to our memory efficient implementations for graph data structures and running-time optimized heuristics, we could even successfully k-anonymize power law distributed social network graphs with > 100000 vertices and $k = 200$ in less than 30 minutes. However, if the solution size is very large, anonymizing and realizing degree block sequences often becomes inefficient and could not be performed within this time. Further improvements in effectiveness of realizing k-anonymous degree block sequences could be achieved by revising the graph construction process. One approach could be to swap selected vertices for previously unselected ones during the graph construction, if it fails due to existing edges. Furthermore, improving Step 2 of the graph construction process by a more advanced local search heuristic might also improve chances on realizability. Least, solving the graph construction problem by reformulating the (block) degree sequence realization problem as an f-FACTOR problem (see [Har+13] for details) could be implemented for realizing k-anonymous degree sequences as a supergraph of the input graph. The f-FACTOR problem can be solved in polynomial-time. Additionally, an instance is a yes-instance, if and only if the (block) degree sequence is realizable as a supergraph of the input graph.
As for many instances the computed k-anonymous degree block sequences could not be realized using any combination of strategies, the degree block sequence anonymization algorithm of [HHN14] might be enhanced such that it validates those solutions as not realizable.

References

[BDK11] Lars Backstrom, Cynthia Dwork, and Jon Kleinberg. "Wherefore Art Thou R3579X?: Anonymized Social Networks, Hidden Patterns, and Structural Steganography." In: *Commun. ACM* 54.12 (Dec. 2011), pp. 133–141.

[Bre+13] Robert Bredereck, Sepp Hartung, André Nichterlein, and Gerhard Woeginger. "The Complexity of Finding a Large Subgraph under Anonymity Constraints." In: *Proceedings of the 24th International Symposium on Algorithms and Computation (ISAAC '13)*. Vol. 8283. LNCS. Springer, 2013, pp. 152–162.

[EG60] P. Erdős and T. Gallai. "Graphs with Prescribed Degrees of Vertices (in Hungarian)." In: *Math. Lapok* 11 (1960), pp. 264–274.

[Fac13] Facebook. *Facebook Reports Third Quarter 2013 Results*. Oct. 30, 2013.

[Har+13] Sepp Hartung, André Nichterlein, Rolf Niedermeier, and Ondrej Suchý. "A Refined Complexity Analysis of Degree Anonymization on Graphs." In: *Proceedings of the 40th International Colloquium on Automata, Languages and Programming (ICALP '13)*. Vol. 7966. LNCS. Springer, 2013, pp. 594–606.

[Hay+07] Michael Hay, Gerome Miklau, David Jensen, Philipp Weis, and Siddharth Srivastava. *Anonymizing Social Networks*. Tech. rep. SCIENCE, 2007.

[HHN14] Sepp Hartung, Clemens Hoffmann, and André Nichterlein. "Improved Upper and Lower Bound Heuristics for Degree Anonymization in Social Networks." In: (2014).

[LT08] Kun Liu and Evimaria Terzi. "Towards identity anonymization on graphs." In: *Proc. ACM SIGMOD International Conference on Management of Data*. SIGMOD '08. ACM, 2008, pp. 93–106.

[MW04] Adam Meyerson and Ryan Williams. "On the Complexity of Optimal K-anonymity." In: *Proceedings of the Twenty-third ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*. PODS '04. ACM, 2004, pp. 223–228.

[SS98] Pierangela Samarati and Latanya Sweeney. "Generalizing Data to Provide Anonymity when Disclosing Information (Abstract)." In: *Proceedings of the Seventeenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*. PODS '98. ACM, 1998, pp. 188–.

[Swe02] Latanya Sweeney. "K-anonymity: A Model for Protecting Privacy." In: *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.* 10.5 (Oct. 2002), pp. 557–570.

Technische Universität Berlin
Fakultät IV – Elektrotechnik und Informatik
Institut für Softwaretechnik und Theoretische Informatik

Name: Hoffmann, Clemens
Matrikelnummer: 332772

Eidesstaatliche Erklärung zur Bachelor-Arbeit

Hiermit erkläre ich, dass ich die vorliegende Arbeit

Graph Degree Anonymization: Lower Bounds and Heuristics

selbstständig und eigenhändig sowie ohne unerlaubte fremde Hilfe und ausschließlich unter Verwendung der aufgeführten Quellen und Hilfsmittel angefertigt habe.

Datum: _____ Unterschrift: _____