k-Anonymization by Freeform Generalization

Katerina Doka
NTUA, Greece

Mingqiang Xue
I^{2}R, Singapore

Dimitrios Tsoumakos
Ionian University, Greece

Panagiotis Karras
Skoltech, Russia

ABSTRACT

Syntactic data anonymization strives to (i) ensure that an adversary cannot identify an individual's record from published attributes with high probability, and (ii) provide high data utility. These mutually conflicting goals can be expressed as an optimization problem with privacy as the constraint and utility as the objective function. Conventional research using the k-anonymity model has resorted to publishing data in *homogeneous generalized groups*. A recently proposed alternative does not create such cliques; instead, it recasts data values in a *heterogeneous* manner, aiming for higher utility. Nevertheless, such works never defined the problem in the most general terms; thus, the utility gains they achieve are limited. In this paper, we propose a methodology that achieves the full potential of heterogeneity and gains higher utility while providing the same privacy guarantee. We formulate the problem of maximal-utility k-anonymization by *freeform generalization* as a network flow problem. We develop an *optimal* solution therefor using Mixed Integer Programming. Given the non-scalability of this solution, we develop an $O(kn^2)$ Greedy algorithm that has no time-complexity disadvantage vis-á-vis previous approaches, an $O(kn^2 \log n)$ enhanced version thereof, and an $O(kn^3)$ adaptation of the Hungarian algorithm; these algorithms build a set of k perfect matchings from original to anonymized data, a novel approach to the problem. Moreover, our techniques can resist adversaries who may know the employed algorithms. Our experiments with real-world data verify that our schemes achieve near-optimal utility (with gains of up to 41%), while they can exploit parallelism and data partitioning, gaining an efficiency advantage over simpler methods.

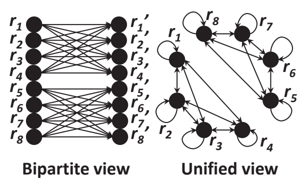
1. INTRODUCTION

The imperative to protect the privacy of individuals [27] requires that a certain *privacy guarantee* be observed when sharing data among agents such as public organizations and private corporations, while disclosing as much information as possible. A popular such guarantee is provided by the k-anonymity model, which requires that the records in a released table should be recast, so that any combination of values on a set of *quasi-identifying attributes* (QI) can be *indistinctly matched to at least k (or none) individuals* therein [24]. This model has been extended and specialized in several forms [23, 20, 7, 6] and other alternatives have been suggested [12, 8]; however, k-anonymity remains a fundamental prerequisite of more advanced models and useful as a stand-alone device in its own right. For example, microtargeted advertising systems in online social networks, even while refraining from selling users' personal information to advertisers, may still inadvertently reveal a user's personal information when an adversary targets an advertisement to a particular user's set of quasi-identifier values [17]. A remedy for this problem requires privacy protections built in by design. Such a protection would be to ensure that an advertiser's targeting criteria never fit less than k user profiles, i.e., to apply the advertising criteria on *k-anonymized* data indeed. Therefore, the k-anonymity model remains topical and relevant in novel settings, and preferable to noise addition techniques in many cases [21, 10].

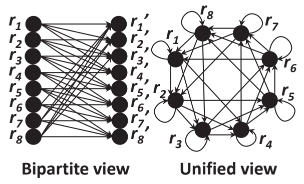
Despite its usefulness in principle, a concern about the applicability of k-anonymity in practice has been caused by a perception that the loss of data utility it engenders would be too large to bear [5], an effect exacerbated as the number of dimensions (QI attributes) grows [2]. However, such loss in utility does not necessarily arise from an inherent drawback of the model itself, but rather from the deficiencies of the algorithms used to implement the model. Indeed, conventional microdata anonymization algorithms have typically departed from the assumption that *all* recast records whose QI values are meant to match the original values of a record t *must* be assigned *identical* QI values to each other [24]; thereby, sanitized records are clustered in disjoint homogeneous groups of the same QI values, called *equivalence classes* [24]. Brickell and Shmatikov first discerned that "*there is no privacy reason*" for this *homogeneity requirement* [5]; they speculated that a strategy using directed acyclic graphs may fare better. In our view, the message to be drawn from [2] and [5] is not a negative, pessimist view that obtaining higher data utility under k-anonymity is impossible, but rather a call for k-anonymization algorithms that do obtain higher data utility by dropping the constraints of previous research. Moreover, we argue that such utility may also be gained *at the expense* of runtime, if a tradeoff between the two emerges. As the anonymization process is an one-off process, some additional runtime is always worth investing for the sake of obtaining higher utility.

This paper provides such algorithms. We observe that some attempts already made in this direction [16, 28, 25, 29, 9] do not define the problem in the most general terms; they introduce superfluous constraints in their solutions or solve the problem by trivially suppressing some values. We keep away from such superfluities and explore the potential to obtain high data utility by value generalization under the k-anonymity model. We define the problem of optimal-utility k-anonymization by value generalization as a network flow problem, a generalization of the assignment prob-

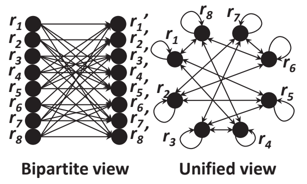
ASIA CCS'15, April 14–17, 2015, Singapore..
Copyright is held by the owner/author(s). Publication rights licensed to ACM.
ACM 978-1-4503-3245-3/15/04 ...$15.00.
http://dx.doi.org/10.1145/2714576.2714590.



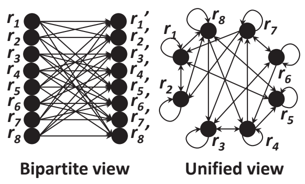
(a) Homogeneous



(b) Ring-based



(c) Heterogeneous Reciprocal



(d) Freeform

Figure 1: Generalization types in graph view

lem on a bipartite graph. To our knowledge, we are the first to formulate this problem in such terms. We develop an exact optimal solution therefor using Mixed Integer Programming. As this solution does not scale, we propose a scalable *Greedy* algorithm, an enhanced version thereof, and a computationally more demanding solution employing the Hungarian algorithm for the Assignment Problem. We also examine whether an off-the-shelf algorithm for the minimum-cost network flow problem can provide a viable solution to our problem. Last, we demonstrate that our techniques can gain an efficiency advantage over simpler methods, when applied in a distributed environment, after partitioning the data.

Our approach differs from preceding research in the *form* of its solutions, which provide better utility, while it provides the same privacy guarantee and recasts data values in the same syntactic way as previous research. A recasting of tuples can be represented by a directed graph, the *generalization graph* [29], that shows how the values of original records match those of anonymized ones. In the *bipartite view* of the graph, an edge from the vertex standing for an original record, r_i, to the one standing for a recast record, r'_j, indicates that the QI values of r_i are included in (*match*) those of r'_j. In the *unified view*, a single vertex represents both the original record r_i and its recast form r'_i.

Figure 1(a) shows the kind of generalization graph constructed by conventional k-anonymization algorithms obeying the homogeneity requirement [24, 19, 15]. In the bipartite view, the partitioning forms two disconnected complete subgraphs of four vertices in each side (i.e., two $K_{4,4}$ *bicliques*), hence obeys 4-anonymity. These subgraphs correspond to the *equivalence classes* formed by those methods; in the unified view, they appear as complete digraphs with self-loops. Previous works [28, 29] eschewed the redundant homogeneity requirement so as to achieve higher utility; still, they resorted to another superfluous requirement, namely that the generalization graph be a *ring*: a cyclical order of vertices is defined, and each vertex matches its predecessors and/or successors over this order. Figure 1(b) shows such a graph.

We propose that homogeneity be eschewed without introducing any other constraint in its place. A corollary of homogeneity is *reciprocity* [29]: when record r_i matches the recast form r'_j of another record r_j, then r_j matches r'_i too; thus, the generalization graph is *symmetric*. To illustrate the distinction between the two, Figure 1(c) shows a generalization graph that is *reciprocal* (records match each other mutually), but *heterogenous* (no record has the same matchings as another). Going further, we can eschew reciprocity too, and aim to construct an *entirely unconstrained* generalization graph that maximizes utility by value generalization. To our knowledge, our work is the *first* to define this problem in such terms. A *freeform* generalization is illustrated by the graph in Figure 1(d).

The advantages of our approach are illustrated by the example in Table 1. The top table presents the values of eight tuples on QI attributes *Age* and *Salary*. By our method, these tuples are anonymized as in the bottom left table; each tuple is recast to a *range* of values, so as to be compatible with, or *match*, three original tuples, and vice versa, as the bottom right table shows; this property

ID	t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7
Age	59	57	39	28	41	37	40	53
Salary	25	27	47	41	20	59	35	34

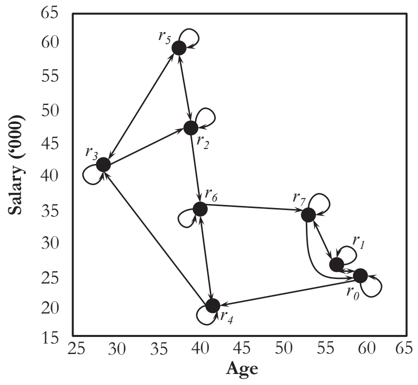
ID	**Age**	**Salary**
t'_0	53-59	25-34
t'_1	53-59	25-34
t'_2	28-39	41-59
t'_3	28-41	20-59
t'_4	40-59	20-35
t'_5	28-39	41-59
t'_6	39-41	20-47
t'_7	40-57	27-35

Original	Matches	Anon/zed	Matches
t_0	t'_0, t'_1, t'_4	t'_0	t_0, t_1, t_7
t_1	t'_0, t'_1, t'_7	t'_1	t_0, t_1, t_7
t_2	t'_2, t'_5, t'_6	t'_2	t_2, t_3, t_5
t_3	t'_2, t'_3, t'_5	t'_3	t_3, t_4, t_5
t_4	t'_3, t'_4, t'_6	t'_4	t_0, t_4, t_6
t_5	t'_2, t'_3, t'_5	t'_5	t_2, t_3, t_5
t_6	t'_4, t'_6, t'_7	t'_6	t_2, t_4, t_6
t_7	t'_0, t'_1, t'_7	t'_7	t_1, t_6, t_7

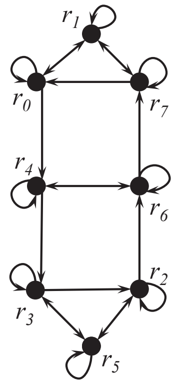
Table 1: Example data anonymized by our method

is called 3-regularity. This property and a randomization scheme guarantee that each original tuple has three *equiprobable* identities [28]; thus, k-regularity is a sufficient condition for k-anonymity.

Figure 2(a) presents the data of Table 1 in a 2d coordinate system where the x-axis stands for *Age* and the y-axis for *Salary*. Each tuple t_i is represented as a point r_i in this coordinate system (shown by a black circle in the figure). An arrow from r_i to r_j denotes that r_i matches the anonymized tuple for r_j. The matching relationships in Table 1 are thus shown in Figure 2(a). For clarity, we present the same matchings in pure (unified) graph form as well, without positioning points by their coordinates, in Figure 2(b).



(a) Graph in 2d plane



(b) Pure graph

Figure 2: Example solution

We reiterate our main contributions as follows:

- We formulate the optimal-utility k-anonymization problem as a network flow problem.
- We provide an exact optimal Mixed Integer Programming-based solution.
- We offer a collection of heuristic solutions.
- We demonstrate experimentally that our techniques achieve significantly lower information loss than previous methods.
- We show our techniques can achieve near-optimal utility.
- We apply our techniques in a parallel-computing environment, gaining an efficiency advantage over previous work.

2. RELATED WORK

The k-anonymity model calls for recasting data so that each original record have *at least k equally probable* matches among recast records. This effect is achieved by a *syntactic* transformation, reducing the *precision* of the data, but not its *accuracy*; the published data is truthful, albeit less exact [19, 5]; a *k-anonymization algorithm* aims to bring the data to a form that abides by the given privacy condition via syntactic transformations, while introducing small inexactness [15]. This type of transformation comes in contrast to anonymization by *perturbation*, which introduces errors in the data; as perturbation-based transformations provide no information on how much a given record has been perturbed, they limit the purposes such data can be useful for [19].

2.1 Striving for Data Utility

Past research has expressed concerns about the usefulness of the k-anonymity model, due to a perception that it engenders high loss of data utility due to the imprecision introduced. Aggarwal [2] noted that, under conventional approaches for k-anonymization, it is difficult to anonymize high-dimensional data "without an unacceptably high amount of information loss". Brickell and Shmatikov [5] went further to claim that data utility is necessarily almost completely destroyed for "modest privacy gains". This claim was re-examined by Li and Li [22]; the point made in [22] amounts, in effect, to the statement that the *privacy loss* and *utility gain* involved in publishing anonymized data are *incommensurable*; they do not share a common standard of measurement, as [5] assumed; *specific* knowledge has a larger impact on privacy, while *aggregate* information has a larger impact on utility [22].

Still, an observation made in [5] regarding classic k-anonymization algorithms is valid. Such algorithms have raised a *homogeneity requirement*, by which recast records form groups of at least k records, such that all records in a group have the same QI values and are hence interchangeable with each other. Brickell and Shmatikov correctly observed that this requirement is redundant, and called for algorithms that drop it to provide improved utility.

Some attempts have been made in this direction [16, 28, 25, 29], yet retain redundant constraints and superfluities in their solutions. Gionis et al. [16, 25] suggested the model of k-concealment, which guarantees that an original record is associated with at least k recast ones, without the homogeneity requirement. However, the $O(kn^2)$ *agglomerative* algorithm in [16, 25] goes through a series of steps that perform *superfluous generalizations*, introducing extra information loss. Besides, as observed in [28] and [25], k-concealment is a weaker guarantee than k-anonymity, as it does not ensure that each association is equiprobable. For example, the graph in Figure 3 satisfies 2-concealment: each record, on both sides of the graph (in bipartite view), has at least two matches, each of whom participates in a complete assignment (perfect matching). However, the matches are *not equiprobable*: some of them (shown with bold lines) participate in three out of four possible assignments, while others (shown with light lines) participate in only one.

Wong et al. [28] studied the conditions for equiprobable associations, and proposed a technique that achieves k-anonymity without the homogeneity requirement. Yet, even while dropping that redundant requirement, [28] does not entirely break its shackles: it still requires that original and recast records relate to each other by a *fixed* motif, *ring generalization*, thus substituting one superfluous constraint for another. Furthermore, [28] does not apply ring generalization on a complete data set, but only within piecemeal partitions obtained via a homogeneous partitioning; the utility gains achieved are primarily due to this partitioning, not due to ring generalization. Xue et al. [29] adopt ring generalization as an element of a technique for anonymizing sparse high-dimensional data; they apply the motif on complete data sets with good results, yet still adhere to that fixed motif. The anonymization algorithms in [28, 29] consist of two components: (i) a component catering to data utility, aiming to produce a data transformation of low information loss by building a ring, and (ii) a component that caters to privacy, aiming to guarantee the equiprobability of k produced associations, via randomization. Even though (i) has sub-quadratic time complexity, the randomization in (ii) brings in an $O(kn^2)$ component. We argue that, since a quadratic time complexity is invested for the sake of privacy anyway, this time complexity budget should also be employed for the sake of utility as well.

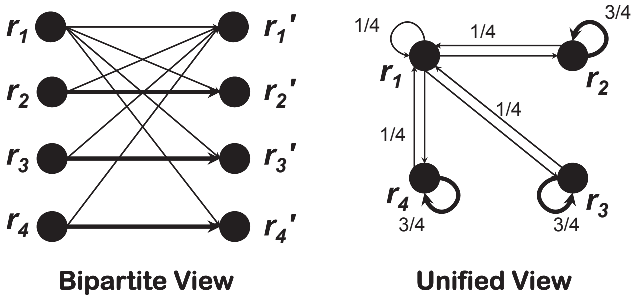


Figure 3: Graph satisfying 2-concealment, but not 2-anonymity

Wong et al. have shown that, to achieve k-anonymity, it suffices to ensure that each original record r_i has exactly k matches in the published data R' and each anonymized record r_i' also has exactly k matches in the original data R; in other words, each vertex in the generalization graph has outdegree and indegree k, i.e., the graph is *k-regular*. It would also suffice to ensure that the generalization graph *contains* a k-regular subgraph, but then we could also maintain that k-regular subgraph only. All graphs in Figure 1 are 4-regular, hence ensure 4-anonymity. The motif of ring generalization is applied exactly to create a k-regular graph in [28], as in Figure 1(c), imposing an arbitrary constraint of its own.

2.2 Security against Adversaries

Wong et al. [28] show that we can extract k disjoint assignments, i.e., one-to-one correspondences between original and recast records, as subsets of a k-regular generalization graph. For the sake of completeness, we reiterate their argument, following a more concise formulation than the one in [28].

LEMMA 2.1. *In a directed graph G, where each vertex v has the same indegree and outdegree d_v, each edge belongs to a cycle.*

PROOF. Assume there is an edge $e = (u \rightarrow v)$ that does not belong to a cycle. Then u cannot be reached from v. Then we classify vertices in two disjoint groups: A, those that can reach u, with $u \in A$ and B, those that can be reached from v, with $v \in B$, where $A \cap B = \emptyset$. By our assumption, vertices in B have the same number $\sum_{w \in B} d_w$ of incoming and outgoing edges. All outgoing edges should point to a vertex in B, since they cannot not reach u. Yet there is an incoming edge to B, namely e, which originates outside B. Hence, there are at most $\sum_{w \in B} d_w - 1$ incoming edges from B to B, but $\sum_{w \in B} d_w$ outgoing edges from B to B. This is a contradiction of the pigeonhole principle. □

LEMMA 2.2. *In a k-regular directed graph G, each edge belongs to a perfect matching (i.e., a set of cycles incoming to and outgoing from each vertex exactly once).*

PROOF. Consider any edge e. By Lemma 2.1, since all vertices in G have equal indegree and outdegree, e belongs to a cycle. Consider such a cycle C. If C contains all vertices of G, it is a perfect matching itself. If not, then we rewire graph G to G' as follows: for each edge $e = (u \rightarrow v)$ in C, we substitute vertex u by a new vertex u/v, such that the outgoing edges from u/v become the outgoing edges from u and the incoming edges to u/v become the incoming edges to v; each edge in C becomes a self-loop in G'. Since G is k-regular, G' is k-regular as well, with indegree and outdegree k on each vertex, and Lemma 2.1 still applies on G'. We say that all vertices in C have been *matched*. Then, we select an arbitrary unmatched vertex $w \in G'$ and one of its outgoing edges e' at random. We repeat the process, finding a cycle C' for e'. If C' contains a previously matched vertex w', we update the matches of w' according to C'. Note that any previously formed matches that are now destroyed are self-loops in G'; thus, the number of matched vertices is monotonically increasing, with at least one new previously unmatched vertex getting matched at each iteration. Therefore, the process eventually terminates; the resulting set of matches $\mathcal{M}$ forms a perfect matching on the original graph G itself, since each vertex has exactly one incoming and one outgoing edge in $\mathcal{M}$. $\square$

THEOREM 2.3. *In a k-regular directed graph G there exist k disjoint assignments (i.e., perfect matchings).*

PROOF. We pick up a node n_i and an outgoing edge e randomly. By Lemma 2.2, e belongs to an assignment (perfect matching) a. We take the edges of a out of G, to get graph G'. Each node in G' has exactly $k-1$ incoming/outgoing edges, hence G' is $(k-1)$-regular. We repeat iteratively and find k disjoint assignments. $\square$

Any of k such assignments can provide the *putative* identities of recast records, and may be used as a guide when assigning other, non-generalized attributes to them. Hence, a record represented by a vertex in a k-regular generalization graph appears to have k possible identities, each in one of the k possible worlds represented by the k assignments. Yet to satisfy the equiprobability requirement of k-anonymity, we should also ensure that each edge (i.e., match) of a vertex is equally likely to participate in a chosen assignment, in other words, there should be k *equiprobable* disjoint assignments. We achieve this result by selecting one of k disjoint assignments *uniformly at random*. Moreover, in order to resist attacks based on knowledge of some anonymized tuples' identities and/or of the algorithm itself, the particular *set* of k disjoint assignments we choose from is generated by a *randomization* scheme.

In effect, each of the k possible matches for each tuple has, from an adversary's perspective, equal probability to be the true one. An adversary running the same algorithm on the same data would get different results, due to the randomization employed. Using randomization to avert attacks based on knowledge of the algorithm was also recommended in [11]. Wong et al. show how a random assignment can be generated by iteratively extracting cycles from the generalization graph (in unified view) via random walks [28]; Xue et al. improve on the efficiency of this process [29]. As discussed, the time complexity of this randomization scheme is $O(kn^2)$. Our methods utilize the same scheme for the sake of security, thus tolerate adversaries who know the algorithm.

Choromanski et al. [9] recently studied k-anonymization as a matching problem as well, with a view to providing *adaptiveness*, i.e. a different privacy guarantee for each individual. In this context, they have also provided an analysis on the question of security in case a match is already known to an adversary. They conclude that heterogeneous reciprocal generalization (which they call *symmetric b-anonymity*) is as secure as k-anonymity, while freeform (i.e., heterogeneous nonreciprocal) generalization (which they call *asymmetric b-anonymity*) is weaker only against sustained attacks by an adversary who gains knowledge of *some* true matches. In particular, if an adversary knows c true matches, then the security of heterogeneous reciprocal generalization drops to $(k-c)$-anonymity, exactly as for homogeneous generalization. However, the security of heterogeneous nonreciprocal generalization drops to $(k-c-\phi(k))$-anonymity, for a function ϕ that satisfies certain conditions. The root cause of this difference is that, after we delete all nodes adjacent to c edges (true matches) known to an adversary, we are left with a graph that contains a $(k-c)$-regular subgraph in the former case, but not always so in the latter case. All algorithms we suggest in this paper can be tuned to produce symmetric generalizations. Aside of this security analysis, [9] does not provide algorithms to achieve k-anonymity by value generalization; instead, they resort to suppression, substituting some values by stars.

3. DEFINITIONS AND PRINCIPLES

We consider a dataset $\mathcal{D} = (Q, P)$ of n tuples. $Q = \{q_1, \ldots, q_n\}$, where q_i is the quasi-identifier part of tuple i and $P = \{p_1, \ldots, p_n\}$ is the rest of the record, not considered to contain potentially identifying information. Our task is to recast the values of quasi-identifying attributes in Q, producing an anonymized form thereof, $Q' = \{q'_1, \ldots, q'_n\}$. In this *recasting*, we allow the value of q_i on attribute A_j, q_i^j, to be substituted by a *set* of possible values $\mathcal{V}(q_i^j)$; as in previous work [24, 19, 15], for a numerical attribute, we publish a range of values defined by that set, as shown in Table 1, while for a categorical attribute we publish that set itself. We say that (the quasi-identifier part of) an original tuple q_i and a recast tuple q'_ℓ *match* each other when q'_ℓ *could be* a recast from of q_i, i.e., each q_i^j is included in $\mathcal{V}(q_\ell^j)$. The privacy guarantee of k-anonymity [24] is then defined as follows:

DEFINITION 1. *An* anonymized *data set $\mathcal{D}' = (Q', P)$ satisfies* k-anonymity *with respect to the original data $\mathcal{D} = (Q, P)$ iff each original record $q_i \in \mathcal{D}$ matches* at least *k published records in $\mathcal{D}'$, each having, from an adversary's perspective, equal probability (at most $\frac{1}{k}$) to be the* true match *of q_i.*

This guarantee ensures that an adversary knowing the quasi-identifying part of all records, Q, is not able to identify the *true match* of a record q_i with probability higher than $\frac{1}{k}$. We describe a collection of one-to-one matches encompassing a complete set of original and recast records as an *assignment*.

DEFINITION 2. *Given a data set $\mathcal{D} = (Q, P)$ and a recast version thereof, $\mathcal{D}' = (Q', P)$, an* assignment *α from $\mathcal{D}$ to $\mathcal{D}'$ is an one-to-one mapping, $\alpha = \{(q_{i_1}, q'_{j_1}), \ldots, (q_{i_n}, q'_{j_n})\}$, such that each $q_i \in Q$ is mapped to exactly one $q'_j \in Q'$, where q_i* matches *q'_j. In each pair $(q_i, q'_j) \in \alpha$, we say that q_i is the* preimage *of q'_j and q'_j is the* postimage *of q_i. Two assignments α_s and α_t are* disjoint *if $\alpha_s \cap \alpha_t = \emptyset$.*

In order to achieve k-anonymity, we need to ensure that there exist k disjoint assignments from original tuples in Q to recast tuples in Q'. A set of k disjoint assignments defines k distinct matches in Q' for each $q_i \in Q$ and *vice versa*, i.e., k distinct matches in Q for each $q'_i \in Q'$. The net result can be represented by a *generalization graph* [29], as in Figure 1.

DEFINITION 3. *Given a data set $\mathcal{D} = (Q, P)$ and its anonymized version $\mathcal{D}' = (Q', P)$, a* generalization graph *$G = (V, E)$ is a directed graph in which each vertex $v \in V$ stands for an original/anonymized tuple $q_i \in Q$ and $q'_i \in Q'$, and an edge $(v_i, v_j) \in E$ is present iff q_i matches q'_j.*

Our definition corresponds to the *unified* view of such a graph (see Figure 1). In a *bipartite* view, the vertex standing for an original tuple q_i is separate from that standing for its anonymized form q_i'. A set of k disjoint assignments defines (and is defined by) a generalization graph in which each vertex has exactly k outgoing and k incoming edges, i.e., a k-regular generalization graph [29]. By constructing a set of k such assignments, we determine that the set of possible values of a tuple $q_i' \in Q'$ on an attribute A_j, $\mathcal{V}(q_i^j)$, should include those of the tuples in Q *mapped* to q_i'. As discussed in Section 2, once we have a k-regular generalization graph, we can randomly regenerate a set of k disjoint assignments, select one of them *uniformly at random* as the one that defines the *true matches* between $\mathcal{D}$ and $\mathcal{D}'$, and publish any other attributes of our data (i.e., in P) accordingly. We *reiterate* that the random character of this process ensures the equiprobability property of k-anonymity.

Based on the preceding discussion, the problem of k-anonymization is translated to a problem of determining a k-regular generalization graph from original to anonymized tuples, and then generalize the attribute values of each anonymized tuple q_i' so as to include the values of its k matches. We aim to find a generalization graph that achieves low information loss. Previous research [14, 15, 28, 7, 6] has used several variants of a Global Certainty Penalty (GCP) as a measure of information loss. We opt for a similar metric, in which we distinguish between numerical and categorical attributes in a way that reflects the way we publish the data. For a numerical attribute A_j, published as a range, we define the Normalized Certainty Penalty, NCP, for a recast tuple q_i' as follows:

$$NCP_j(q_i') = \frac{u_i^j - l_i^j}{U^j - L^j} \tag{1}$$

where u_i^j (l_i^j) is the largest (smallest) value of attribute A_j in the set of possible values of q_i', $\mathcal{V}(q_i^j)$, (i.e., among the matches of q_i'), and U^j (L^j) is the largest (smallest) value in the domain of attribute A_j. The published ranges prevent the determination of an individual's presence in the data, while they can be used for query processing assuming uniform distribution of values within a range [14, 15]. On the other hand, in case A_j is a categorical attribute, we define the NCP for a recast tuple q_i' as follows:

$$NCP_j(q_i') = \frac{count_j(q_i') - 1}{|A_j| - 1} \tag{2}$$

where $count_j(q_i')$ is the number of distinct values of attribute A_j in $\mathcal{V}(q_i^j)$, and $|A_j|$ is the cardinality of the domain of A_j. A similar metric is employed in [28]. By definition, the NCP obtains values between 0 and 1, where 0 signifies no information loss and 1 signifies the maximum information loss for the attribute in question. Then the GCP for a set of recast tuples Q' is defined as:

$$GCP(Q') = \frac{\sum_{q_i' \in Q'} \sum_j NCP_j(q_i')}{d \cdot |Q'|} \tag{3}$$

where j is the index of any attribute A_j in Q, d is the number of all such attributes, and $|Q'|$ the number of tuples in Q and Q'. Our definition of GCP is the average value of NCP among all attributes and all tuples. We aim to minimize this GCP value, hence the problem of *optimal* k-anonymization calls for satisfying the k-anonymity guarantee with a minimal reduction in the utility of the original data:

PROBLEM 1. *Given a data set $\mathcal{D} = (Q, P)$,* transform *$\mathcal{D}$ to an anonymized form $\mathcal{D}'$ that satisfies k-anonymity, such that $GCP(Q')$ is minimized.*

4. OPTIMAL SOLUTION

The methodology proposed in [28] and adopted in [29] creates a fixed k-regular *ring* generalization graphs, without taking into account the actual data values involved. The chief contribution of [28] lies in the randomized process that extracts k disjoint assignments from those graphs in a secure fashion, while that of [29] lies in devising a total order over the records that yields good utility after the fixed graph pattern is applied on it for sparse set-valued data. However, the information loss incurred by the anonymization process eventually depends on the exact *form* of graph built over the data. Unfortunately, the problem of building a graph that minimizes information loss is not addressed in [28, 29]. As we discussed, [28] uses a fixed-form solution and [29] follows suit by adopting it.

Our contribution lies exactly on this graph construction process. We aim to build the graph in a way that minimizes the information lost by value generalization or achieves a near-minimal value of it; to our knowledge, we are the first to address this problem in such terms. In this section, we examine the possibility for an *optimal* solution that builds a k-regular generalization graph. The construction of such a graph corresponds to selecting the set of edges that define it, out of all the available edges in the *complete* bipartite graph from Q to Q'. Viewed in this manner, our problem is a special case of a network flow problem [3]. In network flow terminology, the characteristics of this special case are outlined as follows:

- Our network is a complete bipartite graph from Q to Q'.
- All n vertices in Q are sources supplying k units of flow, and all n vertices in Q' side are sinks demanding k flow units.
- The flow across each edge can take binary values in $\{0, 1\}$.
- The objective is to minimize our GCP function.

We can formulate this problem using techniques of Mixed Integer Programming (MIP). For a numerical attribute A_j, we employ auxiliary variables u_i^j and l_i^j that stand for the maximum (minimum) value in $\mathcal{V}(q_i^j)$. Let q_i^j be the actual value of original tuple q_i on attribute A_j. Last, let $x(\ell, i)$ be a *binary* variable denoting whether the edge from q_ℓ to q_i' is included in the generalization graph we are building (i.e., whether the values of q_ℓ are included in the possible values of q_i'). Then the relationship between original and recast tuples can be expressed via the following constraints:

$$u_i^j \geq q_\ell^j \cdot x(\ell, i) + (1 - x(\ell, i)) \cdot q_i^j \quad \forall q_\ell \tag{4}$$

$$u_i^j \geq q_i^j \tag{5}$$

$$l_i^j \leq q_\ell^j \cdot x(\ell, i) + (1 - x(\ell, i)) \cdot q_i^j \quad \forall q_\ell \tag{6}$$

$$l_i^j \leq q_i^j \tag{7}$$

In case A_j is a categorical attribute A_j, our formulation is slightly different. We substitute A_j by a set of $|A_j|$ auxiliary binary attributes, denoted as $\mathcal{B}_j$, one for each value in the domain of A_j. Then an original tuple q_i has value 1 in one of these attributes only, and 0 in the others, while a recast tuple q_i' should get value 1 in each auxiliary attribute corresponding to a value in the domain of A_j that is in the set of possible A_j values of q_i', $\mathcal{V}(q_i^j)$. Using h as an index for the *auxiliary* attributes for A_j, it now suffices to employ one auxiliary variable, u_i^h, that stands for the maximum value in $\mathcal{V}(q_i^h)$, which is either 1 or 0. The sum $\sum_{h \in \mathcal{B}_j} u_i^h$ for a given tuple q_i' and attribute A_j denotes the number of distinct values of A_j in $\mathcal{V}(q_i^j)$, i.e., equals $count_j(q_i')$ in Equation 2. Using other notations as before, the constraints can be expressed as:

$$u_i^h \geq q_\ell^h \cdot x(\ell,i) + (1 - x(\ell,i)) \cdot q_i^h \quad \forall q_\ell \tag{8}$$

$$u_i^h \geq q_i^h \tag{9}$$

To the above constraints we should add the constraint representing the k-regularity of the graph:

$$\sum_i x(\ell,i) = k \quad \forall \ell \quad \sum_\ell x(\ell,i) = k \quad \forall i \tag{10}$$

Then, denoting the set of numerical attributes as NA and that of categorical attributes as CA, the objective to minimize the GCP metric translates to the minimization of the following quantity:

$$\sum_i \left\{ \sum_{A_j \in NA} \frac{u_i^j - l_i^j}{U^j - L^j} + \sum_{A_j \in CA} \frac{\sum_{h \in \mathcal{B}_j} u_i^h - 1}{|A_j| - 1} \right\} \tag{11}$$

where we follow the notation in Equations (1) and (2). Our formulation is a Mixed Integer Program, where the variables u_i^j and l_i^j are real-valued, while the edge flows $x(\ell,i)$ are constrained to be binary. In Section 8 we show that this formulation can be run by an MIP Solver for small data. Unfortunately, it is prohibitive on sizeable data sets, as Mixed-Integer Programming is NP-hard [4].

5. THE GREEDY ALGORITHM

Given the impracticability of the optimal solution presented in Section 4 for large data, in this section we set up to design a practicable and efficient algorithm for our problem, aiming to achieve near to optimal data utility. Our strategy starts out from the following observation: Instead of striving to build a k-regular generalization graph over the data at once, we can do so in a sequence of k distinct iterations, adding a single assignment to the graph under construction at each iteration.

Let $G = (S, T, E)$ be a bipartite graph with the vertex set S standing for original tuples (pre-images) and the vertex set T standing for the recast records (post-images) we aim to define, where $|S| = |T| = n$. The assignment selection starts out with G being a complete graph. Initially, the *weight* of each edge $e_{i,j}$ from S_i to T_j, $w_{i,j}$ is defined as the GCP that will be incurred if the tuple q_j at vertex T_j is recast so as to include the tuple q_i at vertex S_i; for brevity, we call this the cost of recasting q_j as $\{q_i, q_j\}$. At each iteration of our algorithm, we aim to find an assignment (i.e., a set of n edges covering all vertices) from S to T that achieves a low total sum of edge weights. After each iteration, the selected edges are discarded from the graph, and the weights of remaining edges are *redefined* so as to reflect the new state of affairs. Thus, a redefined weight $w_{i,j}$ reflects the *increase* of GCP that will be incurred if we extend the set of possible values of tuple q_j at T_j to include the values of tuple q_i at S_i (i.e., if we recast q_j as $\{q_i, q_j\}$). In effect, at each iteration we attempt to increase the total GCP as little as possible. After k iterations, a k-regular generalization graph is constructed. In fact, the first iteration is redundant, since the self-matching assignment, having zero information loss, is chosen by default. Thus, there are $k - 1$ iterations that matter.

We now discuss the details of assignment selection at each iteration. We sequentially process all vertices in S. For each such $S_i \in S$ we select the edge $e_{i,j}$, matching it to a $T_j \in T$, that has the minimum weight $w_{i,j}$. In other words, we greedily match each q_i to the q_j that incurs the least GCP increase. Thereafter, we omit S_i from S and its chosen match T_j from T. This $O(n^2)$ process terminates when all pre-image vertices in S have been matched, and hence all post-image vertices in T have been used.

Nevertheless, the termination of the process outlined above is not guaranteed. Given that at each iteration the degree of each vertex is reduced by one, at the ℓth iteration, our algorithm works on an incomplete bipartite graph where each pre-image in S connects to $n - \ell + 1$ vertices of T, and vice versa, i.e., on an $(n - \ell + 1)$-regular bipartite graph. While it is always possible to extract an assignment from such a graph, the process outlined above may encounter a dead-end, in case all $n - \ell + 1$ possible matches of a certain vertex S_i have already been matched to preceding vertices of S and are hence unavailable. To resolve this problem, when we encounter such a dead-end, we perform a *backtracking* process as follows.

Algorithm 1: Greedy Algorithm Iteration

```
   Data: A weighted bipartite graph $G = (S, T, E)$
   Result: An assignment $\mathcal{A}$ with weight close to minimum
1  while $S \neq \oslash$ do
2      select next vertex $S_i \in S$;
3      if $\nexists$ available vertex in T connected to $S_i$ then
4          find $S_{i-x}$ matched to $T_j$ such that $e_{i,j}$ and $e_{i-x,m}$ are
           available;
5          substitute $e_{i-x,m}$ for $e_{i-x,j}$;
6      else
7          select $T_j$ such that $w_{i,j}$ is the minimum of all edges incident
           to $S_i$;
8      $S = S - S_i, T = T - T_j$;
9      Add $e_{i,j}$ to $\mathcal{A}$;
10 return $\mathcal{A}$;
```

Assume that a dead-end is encountered when processing vertex S_i in the ℓth iteration, i.e. there exists no available match between S_i and any remaining vertex of T. Then we backtrack to vertex S_{i-1}, which has been already matched to a vertex $T_j \in T$ by edge $e_{i-1,j}$, and check whether two edges as follows are available:

1. The edge $e_{i,j}$, so that T_j can be assigned as a match to S_i.
2. Any edge $e_{i-1,m}$ between S_{i-1} and any vertex $T_m \in T$, so that S_{i-1} can obtain another available match in T instead.

In case such available edges exist, we add edge $e_{i,j}$ to the constructed matching and substitute $e_{i-1,j}$ by $e_{i-1,m}$ (in case more than one T_m are available, we select the one of minimum $w_{i-1,m}$). Otherwise, backtracking continues with vertex S_{i-2}, and goes on until it finds an eligible candidate S_{i-x}. A pseudo-code for a single iteration of this Greedy algorithm is shown in Algorithm 1.

The backtracking process forces a dead-end vertex S_i to obtain the first available match T_j of a predecessor vertex S_{i-x}. However, while the match of S_{i-x} has been selected as the one of minimum edge weight, such a consideration is not taken into account during backtracking. Therefore, we should better ensure that the vertices in S are examined in an order such that it is likely that neighboring vertices have similar attribute values. To achieve this effect, we first sort the tuples in S by a *lexicographic order* of their attribute values, positioning these attributes from lower to higher cardinality. Putting attributes of lower cardinality at a higher position in this order ensures that large value changes among consecutive tuples are less frequent; for instance, the order $\{\{a, 1\}, \{a, 3\}, \{b, 2\}, \{b, 4\}\}$, obtained by positioning the low-cardinality alphabetic attribute of these four tuples first, is better than $\{\{1, a\}, \{2, b\}, \{3, a\}, \{4, b\}\}$, obtained by positioning the high-cardinality numerical attribute first.

While backtracking offers a way out of the dead-end, we should prove that it can efficiently find an eligible substitution candidate for practical values of k. We start out with the following lemma.

LEMMA 5.1. *In the ℓth iteration of our Greedy algorithm, if we encounter a dead-end while processing the ith vertex, S_i, we can find a previously matched vertex S_y, which can exchange its matching with S_i and obtain an alternative match T_m, among no more than $2 \cdot \ell + i - n - 3$ previously matched vertices.*

PROOF. The status of our graph while processing the i^{th} vertex at the ℓ^{th} iteration is visualized in the matrix of Figure 4. Rows correspond to vertices in S and columns stand for vertices in T. The entry in the cell (a, b) shows the status of edge $e_{a,b}$. A "$\times$" indicates that S_a has been matched to T_b in a previous iteration. Without loss of generality, we arrange the matrix columns so that all post-images in T to which S_i has been matched in the previous $\ell-1$ iterations are gathered in positions $\{T_1, \ldots, T_{\ell-1}\}$. The corresponding cells are labeled with "$\times$" in Figure 4. We do not show other "$\times$" entries as they are inconsequential for the proof, but we keep in mind that there are exactly ℓ–1 "$\times$" entries in each row and each column, corresponding to previously deleted edges. Then, an "$\bigcirc$" indicates that S_a has been matched to T_b in the current, ℓ^{th} iteration. Thus, each of rows $\{S_1 \ldots S_{i-1}\}$ contains exactly one "$\bigcirc$". Besides, each column contains at most one "$\bigcirc$", since a post-image in T can be matched to at most one pre-image in S in the current iteration.

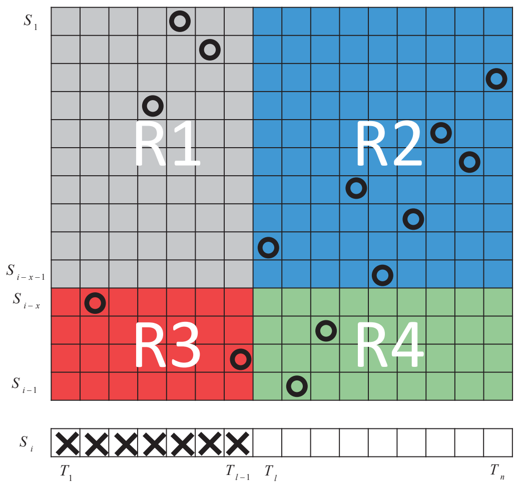


Figure 4: Matrix representation for bipartite matching

An edge from S_a to T_b will be available for matching provided that (i) the corresponding cell is empty; and (ii) there is no "$\bigcirc$" entry in the whole T_b column. Since we encounter a dead-end at vertex S_i, we deduce that each column in $\{T_\ell, \ldots, T_n\}$, for which the cell in the S_i row is empty, contains exactly one "$\bigcirc$" entry, as Figure 4 shows.

Without loss of generality, we gather the rows for *all* x vertices that need to be revisited by our backtracking process in the worst-case scenario at the bottom of the matrix, from S_{i-x} to S_{i-1}. Then our matrix is divided into four disjoint regions, shown by different colors in Figure 4, such that $R1 = \{S_1, \ldots, S_{i-x-1}\} \times \{T_1, \ldots, T_{\ell-1}\}$, $R2 = \{S_1, \ldots, S_{i-x-1}\} \times \{T_\ell, \ldots, T_n\}$, $R3 = \{S_{i-x}, \ldots, S_{i-1}\} \times \{T_1, \ldots, T_{\ell-1}\}$, and $R4 = \{S_{i-x}, \ldots, S_{i-1}\} \times \{T_\ell, \ldots, T_n\}$. The row of a vertex S_y, with $y \in [i-x, i-1]$, that can exchange its match with S_i should satisfy the following two requirements, which correspond to the existence of edges $e_{i,j}$ and $e_{y,m}$ we have seen in our discussion of backtracking:

1. S_y should have been given a match $T_j \in \{T_\ell, \ldots, T_n\}$ in the current iteration that it can transfer to S_i; thus, the row of S_y should have an "$\bigcirc$" entry in $R4$.

2. There should exist a T_m, *neither* already used in the current iteration and *nor* matched to S_y in previous iterations, which S_y can obtain as a substitute for T_j. Such a T_m cannot be in $\{T_\ell, \ldots, T_n\}$, given that all T matches therein have been used already in the current iteration, hence their columns contain "$\bigcirc$" entries, as discussed; it should then be a $T_m \in \{T_1, \ldots, T_{\ell-1}\}$; thus, the row of S_y should have an empty cell in $R3$ whose column does not contain an "$\bigcirc$".

Let w_1, w_2, w_3, and w_4 denote the number of "$\bigcirc$" entries in region $R1$, $R2$, $R3$, and $R4$, respectively. We devise sufficient conditions for the above requirements as follows: Requirement (2) implies that there should exist a column $T_m \in \{T_1, \ldots, T_{\ell-1}\}$ that does *not* contain an "$\bigcirc$" entry. Thus, the total number of "$\bigcirc$" entries in these columns should be *less* than the number of columns themselves. This number of "$\bigcirc$" entries is $w_1 + w_3$, while the number of columns is $\ell - 1$. Thus, it should hold that:

$$w_1 + w_3 < \ell - 1 \tag{12}$$

Furthermore, since each column contains exactly $\ell-1$ "$\times$" entries, and columns in $\{T_1, \ldots, T_{\ell-1}\}$ already contain an "$\times$" entry in row S_i, it follows that each column segment within region $R3$ contains *at most* $\ell-2$ "$\times$" entries. Then any group of $\ell-1$ cells within column T_m in $R3$ will necessarily contain at least one non-"$\times$" entry. Given that column T_m does not contain an "$\bigcirc$" entry either, as Inequality (12) guarantees, it follows that any group of ℓ–1 cells within column T_m in $R3$ will necessarily contain at least one *empty cell*. Thus, as long as $x \geq \ell-1$, Inequality (12) provides that *any set* of $\ell-1$ rows in $\{S_{i-x}, \ldots, S_{i-1}\}$, spanning $R3$ and $R4$, contains a row S_y having an empty cell in $R3$ whose column does not contain an "$\bigcirc$", as Requirement (2) stipulates. It only remains to devise such a set in a way that fulfills Requirement (1) as well, namely guarantees that row S_y *also* has an "$\bigcirc$" entry in $R4$. To that end, if suffices to postulate that $R4$ contains at least $\ell-1$ "$\bigcirc$" entries:

$$w_4 \geq \ell - 1 \tag{13}$$

Since each "$\bigcirc$" entry occupies a different row, Inequality (13) implies that there exists a set of $\ell-1$ rows with "$\bigcirc$" entries in $R4$, while by Inequality (12), one of them will have an empty cell in $R3$ whose column does not contain an "$\bigcirc$" as well. Thus, when both Inequalities (12) and (13) are satisfied, there exists a row S_y with "$\bigcirc$" in $R4$ and an empty entry in $R3$ without a "$\bigcirc$" in its column.

Now we proceed to find what values of x, the number of revisited vertices, render both inequalities true. Since each row in the matrix contains exactly one "$\bigcirc$" and each column in $\{T_\ell, \ldots, \mathcal{T}_n\}$ contains one "$\bigcirc$", it follows that:

$$w_1 + w_2 = i - x - 1 \tag{14}$$
$$w_3 + w_4 = x \tag{15}$$
$$w_2 + w_4 = n - \ell + 1 \tag{16}$$

By Equations (14)+(15)-(16), we get:

$$w_1 + w_3 = (\ell - 1) - (n - i + 1) \tag{17}$$

Since it always holds that $n-i+1 > 0$, Inequality (12) is always true. Then, by Equations (14)-(16):

$$w_4 = n - \ell - i + x + 2 + w_1 \tag{18}$$

Since $w_1 \geq 0$, it follows that:

$$w_4 \geq n - \ell - i + x + 2 \tag{19}$$

Then, to satisfy Inequality (13), we have to ensure that:

$$n - \ell - i + x + 2 \geq \ell - 1 \tag{20}$$

which is equivalent to:

$$x \geq 2 \cdot \ell + i - n - 3 \tag{21}$$

Thus, we can find the desired S_y by examining no more than $2 \cdot \ell + i - n - 3$ previously matched vertices. □

We can now prove the following Theorem:

THEOREM 5.2. *Our Greedy algorithm always resolves the dead-end for $k < \frac{n+3}{2}$.*

PROOF. By Lemma (5.1), in order to resolve a dead-end encountered by the algorithm while processing the i^{th} vertex in the ℓ^{th} iteration, we need to be able to revisit up to $2 \cdot \ell + i - n - 3$ previously matched vertices. In order for so many previously matched vertices to be available, it should be $i > 2 \cdot \ell + i - n - 3$, or $\ell < \frac{n+3}{2}$. Since the algorithm requires k iterations, it can resolve the dead-end in linear time for any $k < \frac{n+3}{2}$. □

In conclusion, our Greedy algorithm works for practical values of k used in real-world settings. Furthermore, the linear-time backtracking process does not affect the $O(n^2)$ complexity of an iteration. Hence, the overall complexity of all k iterations is $O(kn^2)$.

6. THE SORTGREEDY ALGORITHM

The algorithm we presented in Section 5 is greedy in the sense that it makes a greedy choice when it selects the *lightest* available edge of each vertex, while scanning vertices in S sequentially. Nevertheless, this process does not necessarily lead to good edge choices from a global view. For example, assume that edges $e_{i,j}$ and $e_{k,j}$ are both available for pickup, while $w_{i,j} > w_{k,j}$ and S_i is the next vertex to be processed. Then, assuming $e_{i,j}$ is the lightest edge incident to S_i, it will be picked up; thus, $e_{k,j}$ will be rendered unavailable, even though it was a better choice of edge from a global (though still greedy) perspective.

Motivated by this observation, we propose an enhanced greedy algorithm, which we call SortGreedy. The external shell of the algorithm remains the same, i.e., it operates over k iterations, with each iteration striving to select an assignment that brings about a small increase of the total GCP, and edge weights properly redefined among iterations. What differs is the internal edge selection process within each iteration. We outline this process below.

Algorithm 2: SortGreedy Algorithm Iteration

```
   Data: A weighted bipartite graph $G = (S, T, E)$
   Result: An assignment $\mathcal{A}$ with weight close to minimum
1  Sort edges $E$ by ascending weight;
2  while $E \neq \oslash$ do
3      Select $e_{i,j}$ with minimum weight;
4      if $S_i \in S$ and $T_j \in T$ then
5          $S = S - S_i, T = T - T_j$;
6          Remove $e_{i,j}$ from $E$;
7          Add $e_{i,j}$ to $\mathcal{A}$;
8  if $\exists$ unmatched vertices then
9      foreach unmatched vertex $S_i \in S$ do
10         find $S_y$ matched to $T_j$ such that $e_{i,j}$ and $e_{y,m}$ are available;
11         substitute $e_{y,m}$ for $e_{y,j}$ and add $e_{i,j}$ to $\mathcal{A}$;
12 return $\mathcal{A}$;
```

We first sort all edges in E by ascending weight at $O(n^2 \log n)$ cost. Then, instead of scanning a vertex list S, we scan the sorted list of edges instead and try to pick up good edges directly therefrom. For each encountered edge, $e_{i,j}$, we check whether its adjacent vertices, S_i and T_j, are both available for matching. If that is the case, we select $e_{i,j}$ as a match and remove it from E, while also removing S_i from S and T_j from T, as they are no longer available for matching. Otherwise, we proceed to the next edge in the sorted list, until all edges are examined.

As with our basic Greedy algorithm, the above process may not terminate successfully, i.e., it may not have built a perfect matching of n edges after one pass through the edge list; some vertices may remain unmatched even after all edges have been processed. If this is the case, we call a *backtracking* procedure similar to the one outlined in Section 5. We scan the vertex list S, in lexicographic order, so as to detect unmatched vertices; for each such vertex S_i we look for an eligible substitution candidate among its neighbors in the lexicographic order; now we look not only at its predecessors, but at both predecessors and successors, as already-matched vertices can be found anywhere in the lexicographically ordered list. However, the essence of the backtracking process remains the same, hence Lemma 5.1 and Theorem 5.2 still hold. Algorithm 2 presents the basic iteration of this SortGreedy algorithm. As the complexity of an iteration is dominated by the sorting step, the overall complexity of SortGreedy is $O(kn^2 \log n)$.

7. THE HUNGARIAN-BASED ALGORITHM

Both our greedy algorithms work over k iterations, and at each iteration they attempt to find a perfect matching (assignment) that achieves small sum of edge weights (i.e., GCP increase). They follow a heuristic *greedy* logic in solving k local problems instead of the global problem of finding a k-regular generalization graph that minimizes GCP in one go. We have maintained the heuristic logic of k iterations and enhanced the internal greedy algorithm for assignment extraction. Still, the weight minimization problem addressed by this internal process is polynomially solvable - it *is* the Assignment Problem that finds a Maximum (or Minimum) Weight Perfect Matching (MWPM) in a bipartite graph. We can then apply the $O(n^3)$ Hungarian algorithm that finds an *optimal* solution for this problem as the internal process, while maintaining the shell of k iterations. We call the resulting $O(kn^3)$ algorithm Hungarian for brevity. This algorithm remains a heuristic, as it iteratively performs local optimizations. For the sake of completeness, we offer a brief description of the plain Hungarian algorithm, which we apply for minimizing the GCP increase at each iteration.

The Hungarian algorithm was developed by Kuhn [18], Edmonds and Karp [13], and Tomizawa [26]. Consider the graph in Figure 5a, where edge $e_{i,j} = (X_i, Y_j)$ carries a weight $w_{i,j}$. Without loss of generality, we aim to find a perfect matching of maximum weight (MWPM). Such a matching is formed by red edges in Figure 5a, with total weight 16. Given a bipartite graph G with edges E and a matching M, an **augmenting path** is a path starting out from and terminating at nodes not in M, with edges alternating between $E - M$ and M. The path $X_1 \to Y_2 \to X_2 \to Y_3$ in Figure 5c is an augmenting path with respect to the matching M formed by green edges. Such a path can be used to increase the size of a matching M by exchanging the roles of edges in M and not in M; e.g. $M = \{e_{3,1}, e_{2,2}\}$, will be extended to $\{e_{3,1}, e_{1,2}, e_{2,3}\}$ after exchanging edges within the augmenting path.

A **labeling** $\mathcal{L}$ labels each vertex v in G with a weight $\mathcal{L}(v)$; such weights are shown in rectangles in Figure 5a. $\mathcal{L}$ is **feasible** if $\mathcal{L}(X_i) + \mathcal{L}(Y_j) \geq w_{i,j}$ for any edge $e_{i,j}$. An **equality graph** $E_{\mathcal{L}}$ for $\mathcal{L}$ is formed by the set of edges $\{e_{i,j} : \mathcal{L}(X_i) + \mathcal{L}(Y_j) = w_{i,j}\}$. By the Kuhn-Munkres theorem, if $\mathcal{L}$ is feasible and M is a perfect matching in $E_{\mathcal{L}}$ then M is a MWPM for G. We can see that this theorem holds as follows: Since, by definition, *no* matching can have weight more than the total weight of its vertex labels, a matching $M_{\mathcal{L}}$ that establishes the equality achieves the maximum weight. The Hungarian algorithm iteratively finds a maximum matching (i.e. a matching with the most edges) within $E_{\mathcal{L}}$, and extends $E_{\mathcal{L}}$ by adjusting $\mathcal{L}$, until a perfect matching is found within $E_{\mathcal{L}}$.

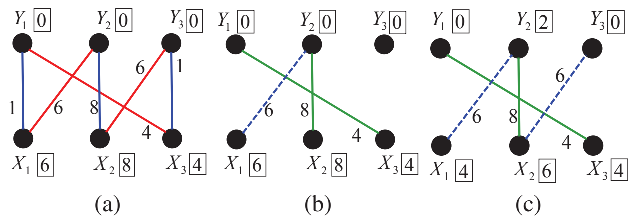


Figure 5: Hungarian Algorithm Illustration

We clarify that, before applying the algorithm to our problem at each iteration, we convert each edge weight $w_{i,j}$ to $-w_{i,j}$, so as to minimize GCP increase. The algorithm starts with an initial feasible labeling $\mathcal{L}$, obtained by assigning the *largest* edge weight at each node on one side, and 0 on the other side. Figure 5a shows such an initial labeling, which creates the equality graph $E_\mathcal{L}$ shown by dotted blue and solid green edges in Figure 5b. We pick a matching $M_\mathcal{L}$ inside $E_\mathcal{L}$, formed by solid green edges, and extend it using augmenting paths found by breadth-first search until no more such path can be found. By Berge's Theorem, a matching $M_\mathcal{L}$ is *maximum* iff it has no augmenting path. When $M_\mathcal{L}$ becomes maximum in $E_\mathcal{L}$, we extend $E_\mathcal{L}$ by reducing the weight of some X_i's by a certain amount σ, so that an edge that was not in $E_\mathcal{L}$ now becomes a part of $E_\mathcal{L}$. In order to ensure the original edges in $E_\mathcal{L}$ are retained after the modification, σ is compensated to the neighbors of X_i. For example, the equality graph in Figure 5b is extended to the one in Figure 5c by reducing the weights of X_1 and X_2 by 2 and increasing the weight of Y_2 by 2, thereby bringing edge $e_{2,3}$ into $E_\mathcal{L}$. Then we extend $M_\mathcal{L}$ in the new $E_\mathcal{L}$. Eventually, if the graph contains a perfect matching, it will be found within $E_\mathcal{L}$ and returned as the MWPM.

8. EXPERIMENTAL EVALUATION

In this section we conduct a thorough experimental study of the algorithms we have introduced in comparison to previous work. To the best of our knowledge, this is the first work to provide experimental results comparing a practical k-anonymization algorithm to an optimal solution. Our study features the following algorithms:

- **NH** The ring-generalization method proposed in [28]. We run this method exactly as it is proposed, with ring generalization applied on the partitions of size between k and $2k-1$. We attempted to apply ring generalization on larger partitions, yet we determined that the best results are obtained when the method runs in its proposed form. This finding confirms that the utility gains achieved by NH are primarily due to the employed partitioning method, rather than due to the ring generalization itself. Runtime measures for NH include the time for partitioning, building rings, and the final randomization step that extracts assignments.
- **k-c** The agglomerative algorithm implementing the k-concealment method in [16, 25]. A randomization step for this algorithm is proposed in [25], aiming to provide security against reverse-engineering the algorithm. Yet this step introduces extra information loss. In order to allow for the best-case scenario for k−c, in terms of both information loss and runtime, we do *not* include this step in our experiments.
- **minCostFlow** The CPLEX solver for the minimum-cost network flow problem, in which the minimized objective function is not GCP, but the sum of edge weights in the original complete graph. We include this so as to check whether an off-the-shelf algorithm can perform well on our problem.
- **MIP** The CPLEX MIP solver running our formulation for an *optimal* solution in Section 4.
- **Greedy** Our Greedy algorithm of Section 5, with a randomization step for assignment extraction as in [28, 29].
- **SortGreedy** Our enhanced greedy algorithm of Section 6, with randomized assignment extraction included.
- **Hungarian** Our Hungarian-based algorithm of Section 7, with assignment extraction by randomization.

Attribute	Cardinality	Type
Age	79	numerical
Gender	2	categorical
Education Level	17	numerical
Marital Status	6	categorical
Race	9	categorical
Work Class	10	categorical
Country	83	categorical
Occupation	51	categorical

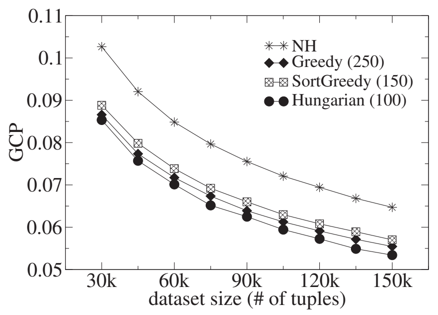
Table 2: The CENSUS dataset

MIP and minCostFlow employ the IBM CPLEX Studio 12.4, invoked by a C++ interface. NH is implemented in C++, while Greedy, SortGreedy, Hungarian, and k−c, are in Java. The methods we compare against have no efficiency disadvantage arising from their implementation environment. All experiments ran on a 24-core Intel Xeon CPU machine @2.67GHz with 48GB RAM running Ubuntu 12.04.1 LTS. We use the CENSUS dataset [1], which contains 500K tuples on 8 attributes as shown in Table 2.

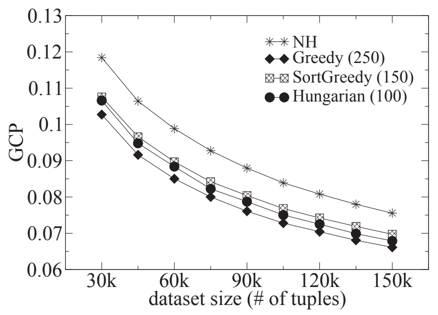
8.1 Evaluation under Partitioning

We commence our experimental study with the following observation: Our methods are configured to run on the *full* data set; nevertheless, the main competing technique, NH, does not do so. We tried to run NH on the full data set, yet the GCP results it achieved were *worse* than those achieved in its default partitioning-based version. This finding indicates that the ring generalization employed by NH may be a liability on large data sets. It is arguably a good idea to apply our methods on a per-partition basis too, for the sake of efficiency. This is what we do in this experiment. We first sort the input data set following a lexicographic order as described in Section 5. Then we divide the data into partitions of equal size P by simply selecting segments of P tuples along the lexicographic order. Furthermore, we use a different partition size for each algorithm, so as to ensure that SortGreedy and Hungarian run at time close to that of Greedy on a single partition. We envisage these algorithms running at a data center offering parallel processing capabilities, with each partition utilizing a different machine. Thus, the runtime of an algorithm is measured as the time for partitioning plus the time for processing a single partition, including the extraction of k random assignments. This configuration allows us to compare the performance of Hungarian, SortGreedy, and Greedy when they are given an equal *time budget* per partition.

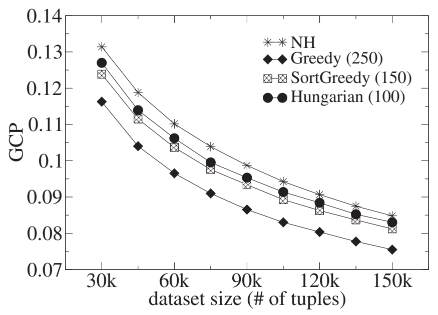
In the case of NH, there is no point of selecting the largest partition in which its core routine can run at time no longer than Greedy, since NH gains *no benefit* by running on larger partitions than the partitions of size from k to $2k-1$ it employs by default. Thus, we stick with the partitions the algorithm inherently uses. However, it would not be fair to assume that all such partitions can be processed in parallel either; that would require far more machines than those utilized by the other methods. Instead, we assume that NH can use the same number of machines *in parallel* as Greedy, i.e. the algorithm that employs the largest size and hence smallest number of partitions among the other three. In other words, we offer to NH the same *parallelism budget* as Greedy. The runtime for NH is measured as the time for running its special partitioning scheme plus



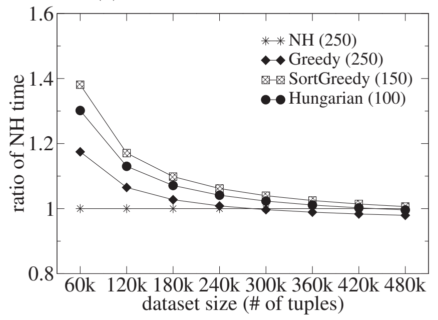
(a) GCP vs. size, $k = 30$



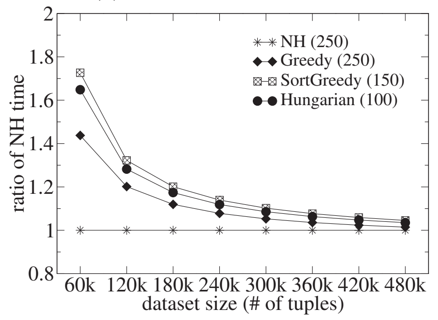
(b) GCP vs. size, $k = 40$



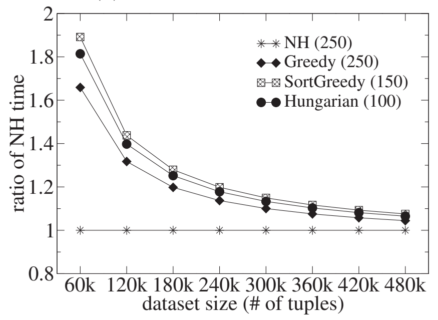
(b) GCP vs. size, $k = 50$



(d) time vs. size, $k = 30$



(e) time vs. size, $k = 40$



(f) time vs. size, $k = 50$

Figure 6: Evaluation under partitioning

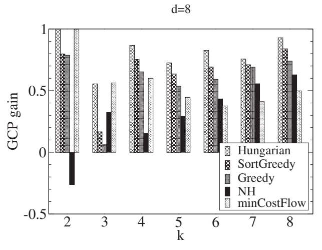
the time for processing as many tuples as in a single partition used by Greedy, including assignment extraction by randomization.

We measure the achieved GCP and runtime, as defined above, for each algorithm. Figure 6 shows our results, as a function of full data size, for three different values of k. As shown in the legends, the employed partition size is 100 tuples for Hungarian, 150 for SortGreedy, and 250 for Greedy. Remarkably, under the same time budget per partition, Greedy achieves better GCP than SortGreedy, while both of them outperform Hungarian as k grows. At the same time, Greedy achieves consistently lower GCP than NH under the same parallelism budget. Figures 6d-f show the runtime of all methods as a ratio of the runtime of NH. Interestingly, our methods approach and eventually surpass the runtime of NH as data size grows. This is due to the fact that NH employs a more demanding partitioning scheme, hence its runtime is eventually dominated by the burden of initially partitioning the data set.

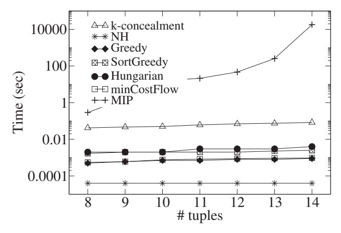
8.2 Comparison to optimal solution

We now examine data sets on which the CPLEX MIP solver can run the optimal solution. This way, we have a chance to assess how close our algorithms arrive to the optimal GCP. To our knowledge, no such experiments have been performed in any preceding experimental study on k-anonymization. We randomly select 100 different data sets of 14 tuples from the CENSUS data set of attribute cardinality $d = 8$ and run all competing techniques for several values of the privacy parameter k. We average the results over the 100 runs. Figure 7a shows our results. The bar charts in the figure show the fraction of the GCP difference between k−c (the worst performer on average) and Optimal each method gains (that is 1 for Optimal, 0 for k-c), vs. k. When a method achieves worse GCP than k−c, its bar shows a negative value. Hungarian achieves consistently the most near-optimal results, followed by SortGreedy at a very close distance. The minCostFlow method achieves, as expected, the optimal result for $k = 2$, since, for this value of k, minimizing the GCP metric is tantamount of minimizing the sum of edge weights selected as the second assignment (the first chosen assignment being the self-assignment). However, as k grows, minCostFlow progressively loses its advantage over other methods. For $k \geq 6$, it fares even worse than NH.

We also measure the average runtime, over 100 runs, as a function of the number of tuples, for $k = 6$. Figure 7b presents our results on logarithmic time axes. * These results show that the runtime of our three polynomial-time algorithms grows modestly as a function of data size, while they are positioned between those of NH and k−c; we will come back to this result later with much larger data. On the other hand, the runtime of the CPLEX MIP Solver grows inexorably, as shown in Figure 7b; running this solver on larger data is a prohibitive task. Therefore, MIP will not be featured in our subsequent experiments. The usefulness of running MIP was in leading us to the finding that our polynomial-time algorithms provide near-optimal solutions indeed.



(a) GCP Gain vs. k, $d = 8$

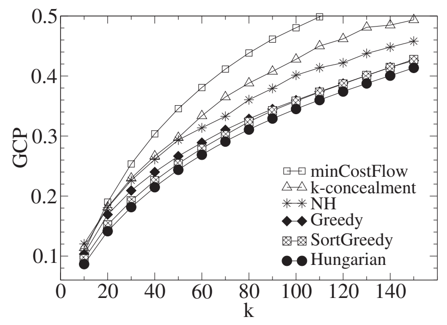


(b) runtime vs. size, $k = 6$, $d = 8$

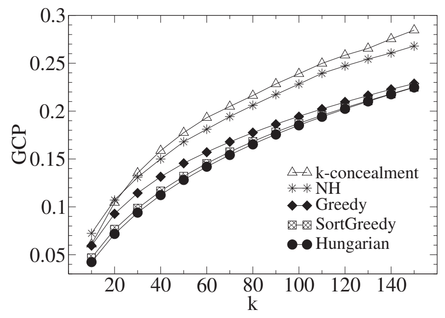
Figure 7: GCP Gain and Runtime on small data

8.3 Effect of k

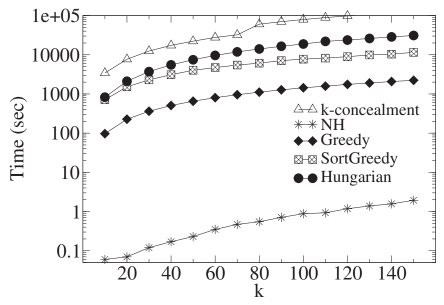
Next, we study the effect of the k parameter on the compared methods, on data sets of 1k and 10k tuples from the CENSUS data set of dimensionality $d = 8$. Figure 8 presents our results, with k ranging from 10 to 150. These results reconfirm the superior performance of our methods in terms of information loss. Remarkably, the GCP divergence between our methods and those of previous works is widened as k grows (with up to 41% improvement), while that between our two greedy methods and Hungarian is narrowed. SortGreedy achieves practically the same GCP as Hungarian for



(a) GCP vs. k, 1K tuples

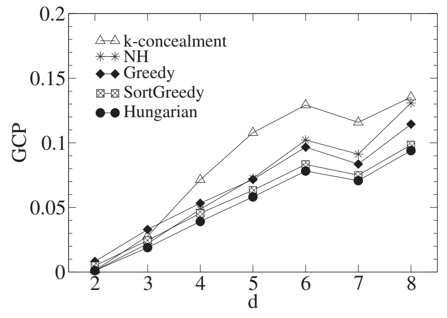


(b) GCP vs. k, 10K tuples

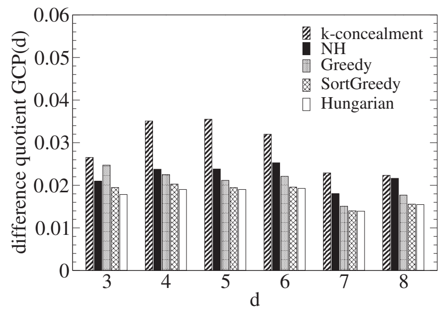


(c) time vs. k, 10K tuples

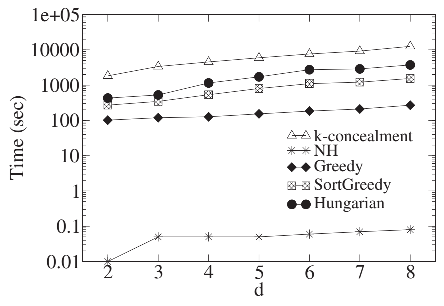
Figure 8: Effect of k



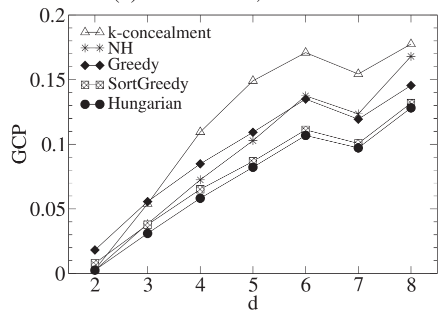
(a) GCP vs. d, $k = 30$



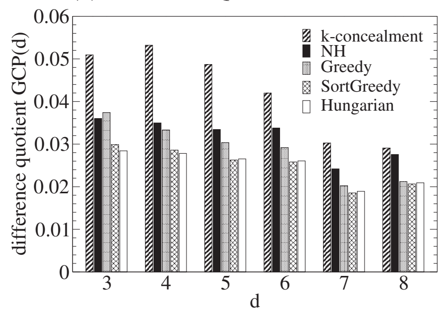
(b) Difference Quotient, $k = 30$



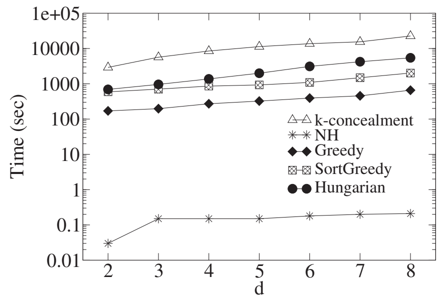
(c) time, $k = 30$



(d) GCP vs. d, $k = 50$



(e) Difference Quotient, $k = 50$



(f) time, $k = 50$

Figure 9: Effect of dimensionality

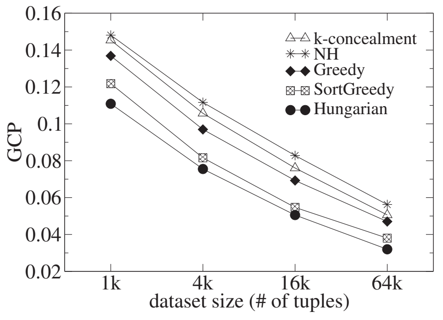
large k. In terms of runtime, our techniques scale equally well as NH with k, while k−c exhibits unstable behavior (Figure 8(c)). We reiterate that paying some extra runtime is worthwhile for the sake of data utility, given that anonymization is an one-off process. The performance of minCostFlow on the 1k data reconfirms that optimizing a non-GCP objective function is not a viable approach. Thus, we omit minCostFlow from subsequent experiments.

8.4 Effect of Dimensionality

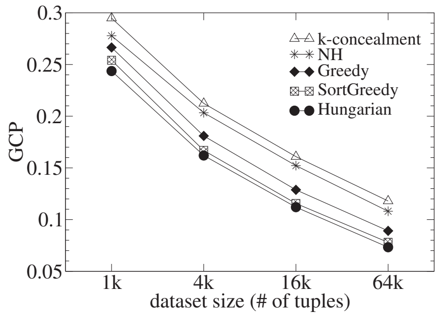
Next, we study the effect of dimensionality on each of the competing techniques. We select the first 10K tuples from the CENSUS data set, and examine the performance of our algorithms as a function of the number of selected attributes d, letting d range from 2 to 8, and setting $k = 30$ and $k = 50$. Figure 9 shows the results. We observe that, not only do our three algorithms achieve better GCP than the competing NH and k−c methods, but they also present better resistance to the curse of dimensionality; the GCP they achieve does not deteriorate as severely as that of the other two methods as d grows. To make this effect more visible, we measure the difference quotient (i.e., the slope) of the GCP as a function of d, $\frac{GCP(d)-GCP(2)}{d-2}$. The middle column in Figure 9 presents our results. Remarkably, k−c has the worst behavior with respect to growing dimensionality, with NH coming second worst. On the other hand, our three algorithms exhibit much better dimensionality robustness, with SortGreedy and Hungarian being consistently the best performers. We also present runtime results for this experiment in the third column of Figure 9 in logarithmic time axes. As before, the runtime of our three methods falls between those of NH and k−c. Still, all algorithms scale equally well with growing dimensionality.

8.5 Effect of Size

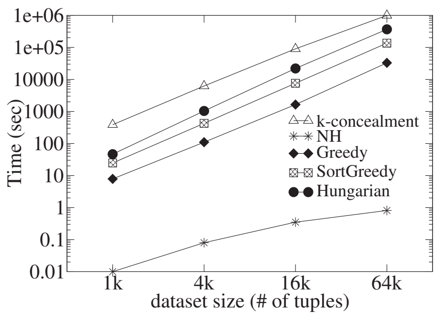
Last, we investigate the scalability of the compared algorithms as the data set size grows. We obtain data sets of exponentially growing size, ranging from 1k to 64k tuples, from the CENSUS data set, with full dimensionality $d = 8$. We present GCP and runtime results in Figure 10, for k values set at $k = 15$ and $k = 50$, using logarithmic scales for all size and time axes. The GCP results present a familiar pattern. Remarkably, our methods consistently outperform NH and k−c, with SortGreedy approaching Hungarian. Concerning the two methods we compare against, it is interesting to note that k−c achieves better GCP than NH for small k. This finding is consistent with the results in Figure 8. The runtime results show that our two greedy methods, Greedy and



(a) GCP vs. size, $k = 15$



(b) GCP vs. size, $k = 50$



(c) time vs. size, $k = 50$

Figure 10: Effect of size

SortGreedy are similarly scalable as k−c, as is expected theoretically. While NH has the same quadratic complexity as those three, its default version which we are running does not actually run the algorithm on the complete data set, but only on partitions thereof. As we saw in Section 8.1, our algorithms have an *efficiency advantage* when given the benefits of data partitioning and parallelism.

9. CONCLUSIONS

This paper casts new light on the k-anonymity privacy model, which remains a prerequisite for more advanced models as well as a useful device in its own right. We treat k-anonymization as a network flow problem, aiming to minimize the information lost by value generalization. While previous works suggested the graph analogy, they either imposed superfluous constraints, or employed value suppression, compromising data utility in both cases. We devise solutions for the most general form of the problem, achieving significantly lower information loss. Conceived in this manner, the problem amounts to building a k-regular bipartite graph that defines an anonymization of high utility. We model an *optimal* solution using Mixed Integer Programming. Furthermore, we devise a greedy algorithm having the same $O(kn^2)$ time complexity as more restrictive previous solutions, an $O(kn^2 \log n)$ enhancement thereof, and an $O(kn^3)$ solution based on the Hungarian algorithm. Our techniques provide the *same* privacy guarantee as previous research on k-anonymity, as well as security against adversaries reverse-engineering the algorithm. Our experimental study shows that our algorithms achieve near-optimal utility and reliably outperform previous work, while their advantage is enhanced as the data dimensionality grows. We show this advantage applies also in terms of time efficiency when working in a parallel processing environment, after we divide a large data set into partitions.

Acknowledgments

We thank Wai Kit Wong and Arnon Mazza, who shared the codes for NH and k-concealment, respectively, as well as Aristides Gionis and Jonathan Eckstein for fruitful discussions on this topic.

10. REFERENCES

[1] http://www.ipums.org.

[2] C. C. Aggarwal. On k-anonymity and the curse of dimensionality. In *VLDB*, 2005.

[3] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, 1993.

[4] R. E. Bixby, M. Fenelon, Z. Gu, E. Rothberg, and R. Wunderling. Mixed-integer programming: A progress report. In M. Grötschel, editor, *The Sharpest Cut*, chapter 18, pages 309–325. 2004.

[5] J. Brickell and V. Shmatikov. The cost of privacy: destruction of data-mining utility in anonymized data publishing. In *KDD*, 2008.

[6] J. Cao and P. Karras. Publishing microdata with a robust privacy guarantee. *PVLDB*, 5(11):1388–1399, 2012.

[7] J. Cao, P. Karras, P. Kalnis, and K.-L. Tan. SABRE: a Sensitive Attribute Bucketization and REdistribution framework for t-closeness. *The VLDB Journal*, 20(1):59–81, 2011.

[8] R. Chaytor and K. Wang. Small domain randomization: Same privacy, more utility. *PVLDB*, 3(1):608–618, 2010.

[9] K. Choromanski, T. Jebara, and K. Tang. Adaptive anonymity via b-matching. In *NIPS*, pages 3192–3200, 2013.

[10] C. Clifton and T. Tassa. On syntactic anonymity and differential privacy. In *PrivDB*, 2013.

[11] G. Cormode, N. Li, T. Li, and D. Srivastava. Minimizing minimality and maximizing utility: Analyzing method-based attacks on anonymized data. *PVLDB*, 3(1):1045–1056, 2010.

[12] C. Dwork. Differential privacy. In *ICALP (2)*, 2006.

[13] J. Edmonds and R. M. Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *J. of the ACM*, 19(2):248–264, 1972.

[14] G. Ghinita, P. Karras, P. Kalnis, and N. Mamoulis. Fast data anonymization with low information loss. In *VLDB*, 2007.

[15] G. Ghinita, P. Karras, P. Kalnis, and N. Mamoulis. A framework for efficient data anonymization under privacy and accuracy constraints. *ACM TODS*, 34(2):1–47, 2009.

[16] A. Gionis, A. Mazza, and T. Tassa. k-anonymization revisited. In *ICDE*, 2008.

[17] A. Korolova. Privacy violations using microtargeted ads: A case study. In *ICDM Workshops*, 2010.

[18] H. W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1–2):83–97, 1955.

[19] K. LeFevre, D. J. DeWitt, and R. Ramakrishnan. Workload-aware anonymization techniques for large-scale datasets. *ACM TODS*, 33(3):17:1–17:47, 2008.

[20] N. Li, T. Li, and S. Venkatasubramanian. Closeness: A new privacy measure for data publishing. *IEEE TKDE*, 22(7):943–956, 2010.

[21] N. Li, W. H. Qardaji, and D. Su. On sampling, anonymization, and differential privacy or, k-anonymization meets differential privacy. In *ASIACCS*, 2012.

[22] T. Li and N. Li. On the tradeoff between privacy and utility in data publishing. In *KDD*, 2009.

[23] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam. ℓ-diversity: Privacy beyond k-anonymity. *ACM TKDD*, 1(1):3, 2007.

[24] P. Samarati. Protecting respondents' identities in microdata release. *IEEE TKDE*, 13(6):1010–1027, 2001.

[25] T. Tassa, A. Mazza, and A. Gionis. k-concealment: An alternative model of k-type anonymity. *Transactions on Data Privacy*, 5(1):189–222, 2012.

[26] N. Tomizawa. On some techniques useful for solution of transportation network problems. *Networks*, 1:173–194, 1971.

[27] R. Wacks. *Privacy. A very short introduction*, volume 221 of *Very short introductions*. Oxford University Press, 2010.

[28] W. K. Wong, N. Mamoulis, and D. W. L. Cheung. Non-homogeneous generalization in privacy preserving data publishing. In *SIGMOD*, 2010.

[29] M. Xue, P. Karras, C. Raïssi, J. Vaidya, and K.-L. Tan. Anonymizing set-valued data by nonreciprocal recoding. In *KDD*, 2012.