Heterogeneous k-Anonymization with High Utility

Katerina Doka*, Mingqiang Xue†, Dimitrios Tsoumakos‡, Panagiotis Karras§, Alfredo Cuzzocrea¶ and Nectarios Koziris*
*National Technical University of Athens, Greece, {katerina,nkoziris}@cslab.ece.ntua.gr
†I²R, Singapore, xuem@i2r.a-star.edu.sg
‡Ionian University, Greece, dtsouma@ionio.gr
§Skoltech, Russia, karras@skoltech.ru
¶University of Trieste and ICAR-CNR, Italy, alfredo.cuzzocrea@dia.units.it

***Abstract*—Among the privacy-preserving approaches that are known in the literature, *k-anonymity* remains the basis of more advanced models while still being useful as a stand-alone solution. Applying k-anonymity in practice, though, incurs severe loss of data utility, thus limiting its effectiveness and reliability in real-life applications and systems. 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.Conventional approaches rely on a methodology that publishes data in *homogeneous generalized groups*. An alternative modern data publishing scheme focuses on publishing the data in *heterogeneous groups* and achieves higher utility, while ensuring the same privacy guarantees. As conventional approaches cannot anonymize data following this heterogeneous scheme, innovative solutions are required for this purpose. Following this approach, in this paper we provide a set of algorithms that ensure high-utility k-anonymity, via solving *an equivalent graph processing problem*.**

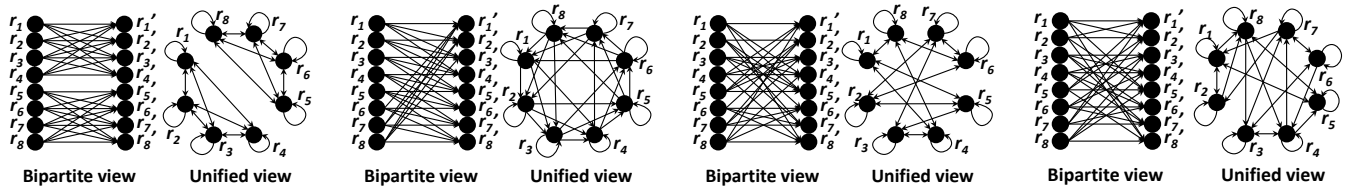
I. Introduction

The imperative to protect the privacy of individuals [1] 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 [2]. This model has been extended and specialized in several forms [3], [4], [5], [6] and other alternatives have been suggested [7], [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 [9]. 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 [10], [11].

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 [12], an effect exacerbated as the number of dimensions (QI attributes) grows [13]. 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 [2]; thereby, sanitized records are clustered in disjoint homogeneous groups of the same QI values, called *equivalence classes* [2]. Brickell and Shmatikov first discerned that "*there is no privacy reason*" for this *homogeneity requirement* [12]; they speculated that a strategy using directed acyclic graphs may fare better. In our view, the message to be drawn from [13] and [12] 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 [14], [15], [16], [17], [18] 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 handle the problem of high-utility k-anonymization by value generalization as an assignment problem on a bipartite graph. To our knowledge, we are the first to address this problem in such terms.

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* [17], that shows how the values of original records match those of anonymized ones. In the *bipartite view* of the graph, an edge



(a) Homogeneous (b) Ring-based (c) Heterogeneous Reciprocal (d) Freeform

Fig. 1. Generalization types in graph view

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 [2], [19], [20]. 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 [15], [17] 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* [17]: 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 I. 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 is called 3-regularity. This property and a randomization scheme guarantee that each original tuple has three *equiprobable* identities [15]; thus, k-regularity is a sufficient condition for k-anonymity.

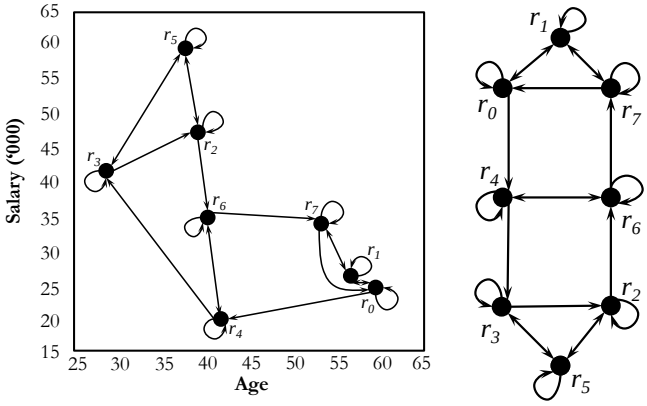
Figure 2(a) presents the data of Table I in a 2d coordinate system where the x-axis stands for *Age* and the y-axis for *Salary*. Each tuple t_i is represented as a point r_i in this coordinate system (shown by a black circle in the figure). An arrow from r_i to r_j denotes that r_i matches the anonymized tuple for r_j. The matching relationships in Table I 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).

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 I
EXAMPLE DATA ANONYMIZED BY OUR METHOD



(a) Graph in 2d plane (b) Pure graph

Fig. 2. Example solution

The full version of this paper appears in [21], where a complete experimental evaluation and analysis of proposed algorithms is also provided.

II. 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 [2], [19], [20], for a numerical attribute, we publish a range of values defined by that set, as shown in Table I, 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 [2] 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* [17], 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 [17]. 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 shown in [22], 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 [23], [20], [15], [5], [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 [23], [20]. 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 [15]. 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.

III. The Greedy Algorithm

The methodology proposed in [15] and adopted in [17] creates a fixed k-regular *ring* generalization graphs, without taking into account the actual data values involved. 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 [15], [17]. As we discussed, [15] uses a fixed-form solution and [17] 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.

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 ℓ^{th} 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 A with weight close to minimum
while S ≠ ∅ do
    select next vertex S_i ∈ S;
    if ∄ available vertex in T connected to S_i then
        find S_{i-x} matched to T_j such that e_{i,j} and e_{i-x,m} are
        available;
        substitute e_{i-x,m} for e_{i-x,j};
    else
        select T_j such that w_{i,j} is the minimum of all edges
        incident to S_i;
    S = S − S_i, T = T − T_j;
    Add e_{i,j} to A;
return A;
```

Assume that a dead-end is encountered when processing vertex S_i in the ℓ^{th} 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.

IV. The SortGreedy Algorithm

The algorithm we presented in Section III 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
Sort edges E by ascending weight;
while $E \neq \oslash$ **do**
  Select $e_{i,j}$ with minimum weight;
  if $S_i \in S$ *and* $T_j \in T$ **then**
    $S = S - S_i$, $T = T - T_j$;
    Remove $e_{i,j}$ from E;
    Add $e_{i,j}$ to $\mathcal{A}$;
if $\exists$ *unmatched vertices* **then**
  foreach *unmatched vertex* $S_i \in S$ **do**
    find S_y matched to T_j such that $e_{i,j}$ and $e_{y,m}$ are available;
    substitute $e_{y,m}$ for $e_{y,j}$ and add $e_{i,j}$ to $\mathcal{A}$;
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 III. 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. 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 $O(kn^2 \log n)$.

V. 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 graph processing 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. Our techniques provide the *same* privacy guarantee as previous research on k-anonymity, as well as security against adversaries reverse-engineering the algorithm.

References

[1] R. Wacks, *Privacy. A very short introduction*, ser. Very short introductions. Oxford University Press, 2010, vol. 221.

[2] P. Samarati, "Protecting respondents' identities in microdata release," *IEEE TKDE*, vol. 13, no. 6, pp. 1010–1027, 2001.

[3] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam, "ℓ-diversity: Privacy beyond k-anonymity," *ACM TKDD*, vol. 1, no. 1, p. 3, 2007.

[4] N. Li, T. Li, and S. Venkatasubramanian, "Closeness: A new privacy measure for data publishing," *IEEE TKDE*, vol. 22, no. 7, pp. 943–956, 2010.

[5] J. Cao, P. Karras, P. Kalnis, and K.-L. Tan, "SABRE: a Sensitive Attribute Bucketization and REdistribution framework for t-closeness," *The VLDB Journal*, vol. 20, no. 1, pp. 59–81, 2011.

[6] J. Cao and P. Karras, "Publishing microdata with a robust privacy guarantee," *PVLDB*, vol. 5, no. 11, pp. 1388–1399, 2012.

[7] C. Dwork, "Differential privacy," in *ICALP (2)*, 2006.

[8] R. Chaytor and K. Wang, "Small domain randomization: Same privacy, more utility," *PVLDB*, vol. 3, no. 1, pp. 608–618, 2010.

[9] A. Korolova, "Privacy violations using microtargeted ads: A case study," in *ICDM Workshops*, 2010.

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

[11] C. Clifton and T. Tassa, "On syntactic anonymity and differential privacy," in *PrivDB*, 2013.

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

[13] C. C. Aggarwal, "On k-anonymity and the curse of dimensionality," in *VLDB*, 2005.

[14] A. Gionis, A. Mazza, and T. Tassa, "k-anonymization revisited," in *ICDE*, 2008.

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

[16] T. Tassa, A. Mazza, and A. Gionis, "k-concealment: An alternative model of k-type anonymity," *Transactions on Data Privacy*, vol. 5, no. 1, pp. 189–222, 2012.

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

[18] K. Choromanski, T. Jebara, and K. Tang, "Adaptive anonymity via b-matching," in *NIPS*, 2013, pp. 3192–3200. [Online]. Available: http://papers.nips.cc/paper/4858-adaptive-anonymity-via-b-matching

[19] K. LeFevre, D. J. DeWitt, and R. Ramakrishnan, "Workload-aware anonymization techniques for large-scale datasets," *ACM TODS*, vol. 33, no. 3, pp. 17:1–17:47, 2008. [Online]. Available: http://doi.acm.org/10.1145/1386118.1386123

[20] G. Ghinita, P. Karras, P. Kalnis, and N. Mamoulis, "A framework for efficient data anonymization under privacy and accuracy constraints," *ACM TODS*, vol. 34, no. 2, pp. 1–47, 2009.

[21] K. Doka, M. Xue, D. Tsoumakos, and P. Karras, "k-anonymization by freeform generalization," in *ASIACCS*, 2015.
[22] W. K. Wong, N. Mamoulis, and D. W. L. Cheung, "Non-homogeneous generalization in privacy preserving data publishing," in *SIGMOD*, 2010.
[23] G. Ghinita, P. Karras, P. Kalnis, and N. Mamoulis, "Fast data anonymization with low information loss," in *VLDB*, 2007.