computers

MDPI

Article

BangA: An Efficient and Flexible Generalization-Based Algorithm for Privacy Preserving Data Publication

Adeel Anjum [1,*] and Guillaume Raschia [2]

[1] Department of Computer Science, Comsats Institute of Information Technology, Park Road Chak Shahzad, 44000 Islamabad, Pakistan

[2] Laboratoire LINA, Ecole Polytechnique, University of Nantes, 44306 Nantes, France; guillaume.raschia@univ-nantes.fr

* Correspondence: adeel.anjum@comsats.edu.pk; Tel.: +92-336-190-2893

Academic Editor: Kartik Gopalan
Received: 25 August 2016; Accepted: 28 December 2016; Published: 4 January 2017

Abstract: Privacy-Preserving Data Publishing (PPDP) has become a critical issue for companies and organizations that would release their data. *k*-Anonymization was proposed as a first generalization model to guarantee against identity disclosure of individual records in a data set. Point access methods (PAMs) are not well studied for the problem of data anonymization. In this article, we propose yet another approximation algorithm for anonymization, coined *BangA*, that combines useful features from Point Access Methods (PAMs) and clustering. Hence, it achieves fast computation and scalability as a PAM, and very high quality thanks to its density-based clustering step. Extensive experiments show the efficiency and effectiveness of our approach. Furthermore, we provide guidelines for extending BangA to achieve a relaxed form of *differential privacy* which provides stronger privacy guarantees as compared to traditional privacy definitions.

Keywords: data privacy; generalization; *k*-anonymity; differential privacy; Bang file

1. Introduction

1.1. Motivation

Data privacy protection is no longer discretionary. The information surge has made the retrieval of public and private information of individuals a part of day-to-day life. Many critical services, e.g., health care, typically gather this information for improving the quality of services; however, given the co-dependency of the Internet and information systems, sensitive data is under the radar of theft and corruption.

Organizations may release their microdata for the purpose of facilitating useful data analysis and research. For example, patients' medical records may be released by a hospital for research purposes. Releasing this kind of data about individuals without risking their privacy is an important problem. To avoid personal identification, many organizations usually remove the uniquely identifying information like *name* or *SSN* from the published data. However, this sanitization of data might not be helpful in guarding the secrecy of given individuals, as it may still be possible to link released records back to their identities by matching some combination of attributes like *age*, *zip code* and *sex*, coined *quasi-identifier* or *linking attributes*. These attributes can be used to infer the sensitive attributes, e.g., *disease* for any individual.

1.2. Basic Requirements

The type of attacks mentioned in Section 1.1 gave rise to the need for robust sanitization methods to publish sensitive individual data keeping their privacy intact. The seminal *k*-anonymization paradigm [1] was proposed to achieve this goal by means of a *generalization model*. Basically, anonymization based on generalization consists of decreasing the accuracy of values from quasi-identifiers. For instance, the 44100 Zip code would become 44xxx and 70 pounds would be said to range between 50 and 80 pounds.

Table 1 provides an example of a public release of six medical records following three-anonymity, i.e., each public record is identical on quasi-identifiers (Age, Zip and Gender) with at least two other records. For instance, records 1, 2, 3 from Table 1 belong to the same equivalence class and are indistinguishable one with each other. The pattern of the class is (Age = (48–62), Zip = 441xx, Gender = *). Similarly, records 4, 5, 6 form the second equivalence class.

Table 1. Three-Anonymous public release.

Id	Age	Zip Code	Gender	Disease
1	(48–62) (62)	441XX (44120)	*(F)	Flu
2	(48–62) (51)	441XX (44190)	*(M)	Flu
3	(48–62) (48)	441XX (44100)	*(M)	HIV
4	(59–77) (59)	444XX (44470)	*(F)	Flu
5	(59–77) (77)	444XX (44420)	*(M)	Gastritis
6	(59–77) (66)	444XX (44420)	*(M)	HIV

1.3. Typical Use Case

The sanitized release must support exploration, analysis and scientific studies. The very first and popular processing of sanitized release is to search and filter tabular data by means of *point queries* and *window queries*. A point query retrieves the records where the predicate matches the exact value e.g., find an employee whose SSN is "3456". In other words, point query relates to an exact point in an *n*-dimensional space. A *window query or range query* retrieves all of the records where some value is between an upper and lower boundary e.g., list of all employees with age between 23 to 25. The result of a window query is a set of points matching a specified range or window. Indeed, regular database records can be geometrically interpreted as points in a multidimensional space where each dimension is a column of the raw table. Point coordinates are then defined by the attribute values. Transformation is obvious for numerical and ordinal variables. Categorical variables could also be equipped with a total ordering, except that without any native ordering, the process is driven by the application domain and background knowledge. Thus, database queries are transformed into queries against a *set of points*.

Once the microdata is sanitized, its records become *hyper-rectangles* in a multidimensional space, where each dimension is a field in the set of quasi-identifiers. For instance, sanitized records from Table 1 are cuboids in the three-dimensional (3D) space (Age, Zipcode, Gender) as shown in Figure 1. As a point query example Q_1, user would filter data to retrieve possible patient's record designated by Age = 62 AND Zip Code = 44120 AND Gender = F. For sanitized release, Q_1 can be expanded for Zip Code and Gender. For instance, Zip Code can be replaced with its value in the increasing levels of its Value Generalization Hierarchy i.e., Zip Code = 44120 OR Zip Code = 4412X OR Zip Code = 441XX OR Zip Code = 44XXX OR Zip Code = 4XXXX OR Zip Code = *. Similarly, Gender = F or *. Consequently, the result set for Q_1 is comprised of 1, 2, 3 from Table 1. Similarly, a window query Q_2 for Zip Code IN [4442X, 4447X] AND Age $\geq$ 50 AND Gender = * is comprised of the result set 4, 5, 6 from Table 1.

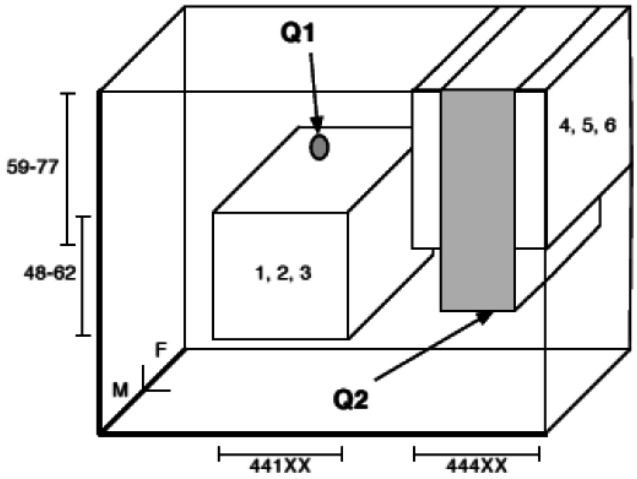

Figure 1. Three-dimensional (3D) spatial representation of the anonymous public release from Table 1 with point query Q_1 and window query Q_2.

To achieve such a querying scenario, the sanitized records are mutually disjoint spatial objects with a *rectangular extent* and window queries are *orthogonal range queries*. Any record that overlaps/lies within the query region is a member of the result set. There exist many efficient algorithms and data structures [2] to compute such orthogonal range queries against the spatial representation of the anonymous database. Furthermore, since any orthogonal range query can be decomposed into several one-dimensional (1D) range queries, it is easy to manage filters on the tabular representation of the public release within a basic spreadsheet or web-client technologies as well. Query Q_2 over Table 1 gives an example of such straightforward decomposition. These practical features are very useful in lots of iterative exploration processes that would support analysis and scientific studies. *Then, we argue that the axis-parallel rectangular coding of anonymous records is a strong requirement for a generic Privacy Preserving Data Publishing PPDP task.*

Other kinds of window queries are defined by the shape of the query region: sphere, half-space, simplex, and polytopes. Sphere range queries, so-called *nearest-neighbor queries*, have been extensively studied, and there also exist efficient algorithms to compute such popular queries especially on rectangular objects. However, none of these range queries satisfies the decomposition property that makes anonymous releases human-friendly under tabular representation. To sum-up the above discussion, we argue that every PPDP task should meet at least the following theoretical and practical requirements in order to be valuable for the end-user:

- Indistinguishability principle—to achieve generalization;
- Mutually disjoint equivalence classes—to preserve quality of the anonymous public release;
- Multidimensional point partitioning—to support point and range queries on the anonymous public release;
- Hyper-rectangular coding of equivalence classes—to allow decomposition of orthogonal range queries.

1.4. Related Work

It is worth noticing that public release with one single equivalence class described on each dimension, by every domain, is obviously k-anonymous ($k \leq n$ the number of records), but it is definitely useless for the end-user. Thus, the main challenge of k-anonymization is to compute a public release where the information loss has been minimized, in the sense of a general criteria by means of popular utility metrics such as *discernibility penalty* [3], *Kullback–Leibler (KL)-divergence* [4] and the certainty metric proposed in [5]. This optimization problem was proved to be NP-hard [6].

Hence, many approximation algorithms have been proposed in the literature since the seminal work of Sweeney [1]. Fung et al. [7] provide a detailed survey on the approximation algorithms for *k*-anonymity. Usually, the Mondrian approach [8] is thought of as the baseline algorithm since it has many interesting properties that we could expect from such algorithms: local recoding and multidimensional partitioning. Mondrian iteratively operates a binary partitioning of the data space until every block contains between k and $2k-1$ data points. Actually, Mondrian builds a *kd*-tree over the raw data and publishes bounding boxes of the leaves as equivalence classes of the anonymous release. Construction has time complexity $O(N\ log(N))$, where N is the number of records in raw data.

Following the geometric representation of the data, Iwuchukwu et al. [9] propose using a bulk-loading implementation of an R^+-tree, one of the most popular spatial access methods for databases, to compute the *k*-anonymous release. It outperforms Mondrian thanks to buffering and efficient bottom-up index construction algorithm, and it scales up to very large data sets. Furthermore, the hierarchical structure of the R^+-tree natively supports $(B^\ell k)$-anonymity for all levels ℓ in the tree, with B the fanout parameter. With an ordered leaf scan, it could support (cK)-anonymity as well, for all c in $\mathbb{N}$. Time complexity remains in $O(N\ log(N))$, and I/O cost for external computation is in $O(\frac{N}{B} log(\frac{N}{B}))$.

Since the R^+-tree bulk-loading algorithm is applied on a set of points rather than a set of spatial objects with an extent, it is actually a variant of a *kd-B*-tree structure where hyper-rectangles have been shrunk to the minimum bounding boxes (MBB) of the subset of points in each equivalence class. It is important to note that a *kd-B*-tree is a bucket-oriented variant of a *kd*-tree, where the fanout of each node is defined by a parameter B that usually fits the disk block size. Therefore, many good features of the R^+-tree approach makes it the reference algorithm for *k*-anonymization up until now.

Many works also proposed point partitioning structures in low dimension (2–3D) for privacy preserving location-based queries [10–13]. In this application domain, privacy is related to instant location of users and queries as well. Popular approaches design an anonymizer that dynamically provides a Cloaking Region to the Location-Based Service. For that purpose, Gruteser et al. [11] implements a *kd*-tree, whereas Mokbel et al. [13] uses a variant of a Point Region (PR) quadtree in *Casper*. Ghinita et al. [10] accommodate partitioning structures from *kd*-tree and *R*-tree to hash a database of Points Of Interest (POI) and answer approximate nearest-neighbor queries in a Privacy Information Retrieval (PIR) approach (*a PIR protocol allows a user to fetch a tuple from a database while concealing the identity of the tuple from a database server*) [14]. They also consider Hilbert space filling curves to map 2D points to single-dimensional data structures like B^+-trees to index POIs. Actually, they argue that their PIR approach is independent from the partitioning structure as far as it provides at most $\sqrt{N}$ buckets within up to $\sqrt{N}$ POIs each. Other work [15] focused on geo-privacy in the sense of privacy-preserving location data publishing. In this context, a space filling curve was also employed to order both data points and POIs on the map. Quad-trees and space filling curves do not scale for higher dimensions, and the latter cannot guarantee non overlapping bounding boxes in the worse case. Furthermore, Arribas et al. [16] and Erola et al. [17] propose anonymizing high dimensional set valued data via microaggregation.

The short review states that every approach to geo-privacy accommodates in memory and implements well-known structure for multidimensional point data partitioning. *k*-anonymity was also studied from the cardinality constraint clustering point of view. On one hand, the anonymization algorithms were proposed [18–20] that achieve good quality, whereas neither they scale up in the size of the data set, nor they meet the basic orthogonal range query requirement since patterns are spheres (centers and radius) of each cluster. On the other hand, many grid clustering techniques ([21,22] for a short excerpt) have been proposed. However, none of them are as fast and scalable as Point Access Methods (PAM) since external storage support and dedicated insert-delete-search operations are missing. Then, PAMs remain the preferred logical structures for the anonymization of very large data sets.

1.5. Synopsis

The *k*-anonymity model proposed by Samarati and Sweeney [23] provides a practical solution for Privacy Preserving Data Publication (PPDP) and has been studied extensively for the last two decades. Anonymization via generalization and/or suppression is able to protect the privacy of individuals, but at the cost of information loss especially for high-dimensional data. This is due to the fact that generalization based *k*-anonymity is impeded by the curse of dimensionality as shown by Aggarwal [24]. Furthermore, in order to achieve an effective generalization, the tuples which share the same set of quasi-identifiers ought be close to each other so that the generalization may not lose too much information. Nevertheless, high-dimensional data forces greater amount of generalization to satisfy basic requirements for *k*-anonymity even for relatively smaller values of the parameter *k*. Hence, it is important to consider deeply the trade-off between privacy and information loss. Thus, the motivating question in this context is *how to minimize the information loss in the course of generalization, especially for high-dimensional data*.

Since the emergence of *k*-Anonymity, many generalization models have been proposed, e.g., ℓ-diversity, *t*-closeness, that tend to overcome the inherent problems with *k*-anonymity. Recently, Differential Privacy (DP) has emerged as a de facto standard in the domain of PPDP. Though DP has originally been proposed in the context of statistical disclosure control, it has gained much anticipated popularity in Privacy Preserving Data Mining (PPDM) and PPDP due to the fact that it promises strong privacy guarantees as compared to the techniques proposed previously. Quite recently, there is an encouraging trend of combining DP style privacy with generalization based approaches. Specifically, the works in [25–27] provide interesting directions to achieve DP for *k*-anonymity based generalization approaches since *Differentially Private k*-anonymous release has an effective privacy vs. utility trade-off.

This article presents *BangA*, a new generalization algorithm that meets generic PPDP features and that offers several new desirable features in regard to many other existing approaches, and especially compared to the R^+-tree based anonymization algorithm [9].

Though BangA generalization algorithm can be extended to achieve any generalization mode, e.g., ℓ-diversity or *t*-closeness, we implement BangA to achieve a *k*-anonymous public release for the following reasons:

- *k*-anonymity is conceptually simple;
- *k*-anonymity does not enforce any constraint on the distribution of sensitive information in a public release [25]. This is one of the main reasons it can be extended to achieve more stronger notions of privacy e.g., *differential privacy* (DP).

The article is organized as follows: Section 2 discusses the main recipe for BangA. Sections 3 and 4 explain how the raw data can be transformed to anonymous public release using BangA. Section 5 evaluates the effectiveness of the proposed approach.

2. General Overview

2.1. Preliminaries

In this section, we provide the basic definitions of some important concepts.

Definition 1. *(Quasi-identifier (from [28])). Consider a set of attributes* $\mathcal{A} = A_1, A_2, ..., A_n$ *sampled from a general population. The set of attributes* $QI_1, ..., QI_w \in \mathcal{A}$ *is said to be a quasi-identifier if these attributes can be used via linking to uniquely identify an individual from the general population.*

Definition 2. *(k-Anonymity from [1,28,29]). Let R be the data set and* $\mathcal{QI}$ *be the set of all quasi-identifiers in it. R satisfies k-Anonymity, if for a record* $t \in R$*, there exist at least* $k-1$ *other records* $t_1, t_2, \ldots, t_{k-1} \in R$ *such that* $t[QI] = t_1[QI] = \ldots = t_{k-1}[QI]$ *for all* $QI \in \mathcal{QI}$*, where* $t[QI]$ *corresponds to the projection of t on the members of QI.*

Definition 3 (X-Equivalence relation $\sim$)**.** *Let R be a table with schema $R(X, Y)$. The X-equivalence relationship $\sim_X \subseteq R \times R$ is defined as: $\forall t, u \in R$, $t \sim_X u \leftrightarrow t[X] = u[X]$.*

For a tuple $t \in R$, the X-equivalence class of t, denoted $[t]_{\sim_X}$, contains similar values on each component of X. In what follows, we refer to this QI-equivalence class as an equivalence class defined by a QI-equivalence relation.

Since the quasi-identifiers are susceptible to linking attacks, the table R is not released directly; it is first processed through a *sanitization mechanism* and then resulting table R^* is published. There exist various sanitization mechanisms in the literature e.g., *generalization, suppression, bucketization*, etc. Since we employ generalization (generalization substitutes a specific value with a more general *less precise* value while preserving the data "truthfulness") as a sanitization mechanism, we provide below a definition of a generalization mechanism to achieve R^*.

Definition 4 (Generalization mechanism $\mathcal{A}$)**.** *Given a micro-data table R, a generalization mechanism is a bijective function $\mathcal{A}$ defined as follows:*

$$\begin{array}{lllll} \mathcal{A}: & R\langle ID, QI, S\rangle & \rightarrow & R\langle ID, QI, S\rangle \\ & \mathsf{I}(R) & \longmapsto & \mathcal{A}(\mathsf{J}(R)) = \mathsf{I}(R^*) = \\ & & & \{\langle t[ID], \nu, t[S]\rangle \mid t[QI] \preceq \nu \ \wedge \ t \in R\}, \end{array} \tag{1}$$

where $J(R)$ is a generalized table of R, and ν is a generalized value of $t[QI]$ according to any pre-defined partial order over QI. For the sake of simplicity, we denote in the following by R the instance $\mathsf{I}(R)$ of R, and by R^* the instance $J(R)$ that is a generalized version of R.

Note that such a $\preceq$ partial order is basically a containment relationship. In addition, $\mathcal{A}(R)$ is not unique since there exist many different ways to generalize $t[QI]$ and the $\mathcal{A}(R)$ enumeration is properly combinatorics. Then, regular approaches try to optimize a utility-based objective function in the generalization mechanism. This is the underlying reason why the k-anonymization based generalization mechanisms have been proven to be NP-hard [4].

We have employed global recoding as proposed by LeFevre et al. [30]. There are several advantages of the global recoding scheme:

- The strategy has conceptual simplicity.
- It is usually tractable to obtain an optimal solution [3].
- Finally, the inferences among the remaining attributes stay uniform with the original data set.

2.2. Overview of BangA

Figure 2 overviews the major steps involved in BangA from raw data to the anonymized release. It is important to mention here that BangA does not provide a straightforward implementation of Bang clustering. It redefines the main steps in bang clustering (e.g., the density index calculation (see Section 4.1)) to achieve high quality public release. It is also worth mentioning that the original definition of Bang clustering is not focused on releasing the data for specific purposes. Thus, we have significantly molded the original Bang clustering to fit into our problem.

BangA consists of a six step process with an overall quasilinear complexity. The complexity of each phase is depicted in Figure 2.

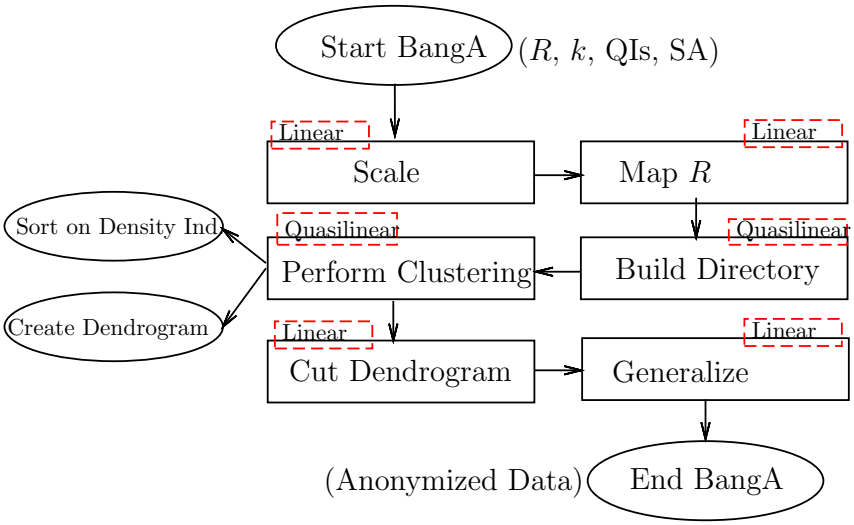

Figure 2. BangA: A big picture.

BangA relies on an index structure, a so-called BANG file [31]. It operates on the axis-parallel space partitioning by means of nested hyper-rectangles rather than disjoint hyper-rectangles only. This singular feature allows for improving expressive power of patterns compared to *kd*-B-trees, including variants like R^+-tree. Consequently, it has a simple splitting strategy as scales are pre-defined over each dimension, thereby substantially increasing the efficiency of the algorithm. BangA starts by mapping the n-dimensional raw data to the unit hypercube $[0, 1]^n$ where the BANG file is going to be defined. The Bang file stores the data points in the underlying space by a tree structure known as *bang directory*. This structure, which is governed by scales, partitions the n-dimensional space into so-called *block regions* or *blocks*. BangA is also able to provide multi-granular anonymous release. It performs a density-based clustering step on blocks of the BANG file to build a dendrogram. Then, a cut in the dendrogram could yield to the required equivalence classes that can be generalized to achieve high quality public releases and provide very flexible settings within one single run.

To sum up, various practically useful features of our approach are given below:

1. Spatial indexing technique—to leverage 30 years of research and experience in effective and efficient external multidimensional partitioning data structures built from very large data sets;
2. BANG file revisited—to accommodate a well-studied logical structure to the generalization problem;
3. Axis–parallel coding of equivalence classes—to ease orthogonal range queries for the end-user;
4. Nested hyper-rectangles—to improve quality of the anonymous public release keeping the axis-parallel coding feature up;
5. Grid-based partitioning—to make the computation faster and to control the privacy requirement by means of knowledge about the data;
6. Density-based clustering of blocks—to enforce quality of the anonymous public release in the process of block merging;
7. Multi-granular anonymization—to allow different settings for the k value with a single run of the algorithm.
8. Methodology for point and orthogonal range queries on non hyper-rectangular tabular data—to support exact match and basic range searching against anonymous public releases into spreadsheets.

Features 1, 3 and 7 are shared with at least the R^+-tree based approach, whereas features 2, 4, 5, 6 and 8 are unique to BangA.

3. From Raw Data to the BANG Directory

3.1. Data Space Partitioning

3.1.1. Mapping n-Tuples to the Unit Hypercube $[0,1]^n$

The very first step of the overall anonymization process in BangA is to map n-dimensional raw data to the unit hypercube $[0,1]^n$ where the BANG file is going to be defined. Records are n-tuples $\langle x_i \rangle_{1 \leq i \leq n}$ over the set of quasi-identifiers, and each field value x_i is an element of an attribute domain D_i. Mapping records $[0,1]^n$ consists of normalizing all the n domains. The operation becomes independent from the kind of variable. For instance, a straightforward linear transformation could be used for interval variables. Ratio and additive variables are also easy to manage. Ordinal variables, isomorphic to the natural numbers, could be handled as well, whereas nominal variables would require more effort to achieve the mapping, especially to define an ordering. Here, the application domain supports the definition of the right ordering. Samet reminds reminders in the introduction of his book [32] Page 479 that "it should be clear that finding an ordering for the range of values of an attribute is not an issue; the only issue is what ordering to use!" Then, any background consideration should be made, e.g., reasoning from domain taxonomies, to help finding the right ordering for categorical values.

The second strong requirement of the user-defined mapping to $[0,1]^n$ is to provide a partitioning of domain D_i within 2^{l_i} ranges, $l_i \geq 1$. The l_i parameter set up the resolution of the dimensional scale that will be used for space decomposition into the BANG file. Similarly, the unit interval $[0,1]$ is partitioned on the ith dimension into equal-sized ranges $[\frac{k}{2^{l_i}}, \frac{k+1}{2^{l_i}}]$, $0 \leq k \leq 2^{l_i}$ that map to the 2^{l_i} ranges from D_i. Since there is no constraint on the mapping from $\Pi_{1 \leq i \leq n} D_i$ to $[0,1]^n$, background knowledge could be incorporated into scales in order for instance to fix undesirable data distribution or to emphasis portions of the data space in the transformation.

3.1.2. Grid Partitioning and Resolution

The n scales define a grid on the multidimensional data space as shown by Figure 3 for a two-dimensional space. Furthermore, as many grid-based structures, the BANG file divides the data space within a hierarchy of regions, where the leaves are the finest grained grid regions corresponding to the grid resolution. The BANG file performs iterative binary partitioning to develop the hierarchy from the entire data space (root) to the grid regions (leaves). Scales are used as partition lines for regular decomposition and the process is *cyclic through the dimensions.*

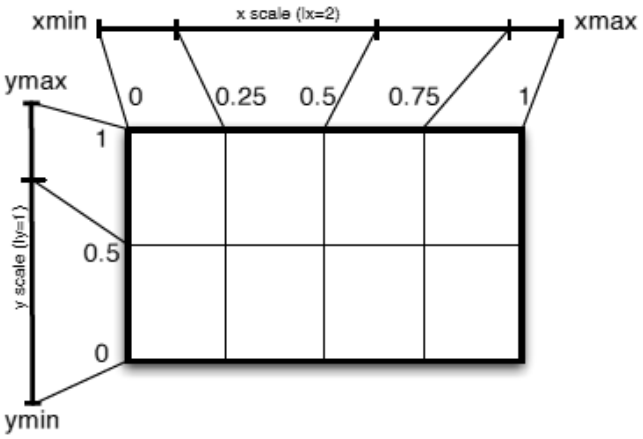

Figure 3. Grid partitioning of the data space by means of regular decompositions following dimensional scales.

Each region in the hierarchy, including grid regions, is identified by a unique pair (r, ℓ), where r is the region number and ℓ is the granularity or level number. A key feature of the BANG file [31] is the region numbering scheme. It relies on a *bitstring* representation of regions that provides very efficient search facilities.

Figure 4 shows the numbering scheme. The outermost region is not given any identifier, whereas each non-root block region is identified by (r, ℓ), with r being the value of a string of binary digits, e.g., **010** is assigned to region $(2, 3)$. Each subspace of a binary partitioning is given value 0 (left/below part) or 1 (right/above part) and regions are identified by the sequence of values of binary partitioning.

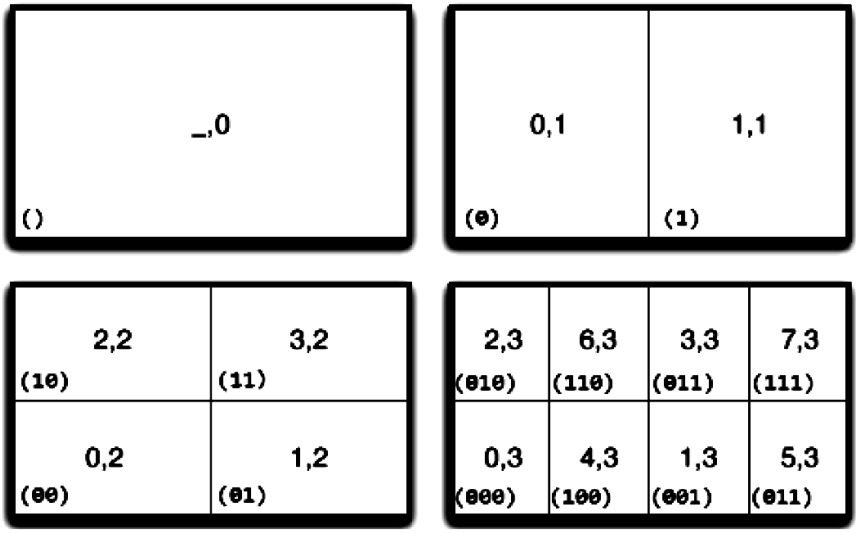

Figure 4. Block region numbering scheme.

3.1.3. About Shape

The BANG file relies on two axioms stated by Freeston [31]:

- The union of all sub-spaces into which the data space has been partitioned must span the data space.
- If two sub-spaces into which the data space has been partitioned intersect, then one of these sub-spaces completely encloses the other.

The second axiom states the existence of *nested regions* in the data space partitioning. Hence, the BANG file removes the requirement that the portions resulting from decomposition of the underlying space that are spanned by a region be hyper-rectangles. The consequence is that the sub-space spanned by a bucket of point data, a so-called *a block*, is a combination of an enclosing region minus a set of enclosed regions. Then, it could be either a hyper-rectangular region or an axis-parallel concave portion of the space or even a disjoint set of sub-blocks.

In the following, we denote by X a hyper-rectangular region of the space given by the regular decomposition of the BANG file. The block enclosed into X is itself denoted by $[X]$. For instance, on Figure 5, the block $[A]$ is assigned to region A and is defined as the sub-space spanned by region A minus regions B, D and G. Only the innermost regions, e.g., C, E, F and G in Figure 5, coincide with their corresponding blocks $[C]$, $[E]$, $[F]$ and $[G]$, respectively. Otherwise, the general definition of a block is as follows:

$$[X] = X - \bigcup_{Y \in \mathcal{H}_X} Y, \tag{2}$$

where $\mathcal{H}_X$ is the set of pairwise disjoint X-enclosed regions at the first level.

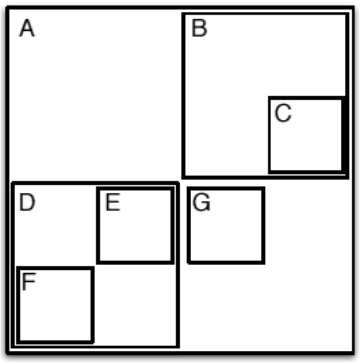

Figure 5. Example of a two-dimensional (2D) data space partitioned with the BANG file into seven nested block regions A, B, C, D, E, F, G.

For the sake of simplicity, we also use $[X]$ to denote the data bucket from which points lie into the sub-space spanned by block $[X]$. Thus, $[X]$ is a subset of points and a complex shape as well, depending on the contextual meaning and without any ambiguity. The advantages of the block definition compared to hyper-rectangle-based *kd*-B-tree structures are a better observation of inherent clusters into data and also a higher filling rate of buckets. A building algorithm as described by Freeston [31] guarantees the balance among buckets by redistribution, thereby making way for clustered value sets.

3.2. Mapping Scheme

Both insertion and searching into the BANG directory require mapping data point coordinates to a block where the point lies by the way of the enclosing region number. To this end, the BANG file defines a set of hash functions [31] from data point coordinate $\langle x_i \rangle_{1 \leq i \leq n}$ to region number r at scaling level k_i, by means of enclosing region coordinate $\langle d_i^{k_i} \rangle_{1 \leq i \leq n}$

$$d_i^{k_i} = \frac{\lfloor 2^{l_i}.x_i \rfloor}{2^{l_i - k_i}}, 0 \leq k_i \leq l_i. \tag{3}$$

Then, the convenient bit-string representation of region numbers allows for concatenating dimensional $d_i^{k_i}$ coordinates at levels k_i to one single $(r, \ell = \sum_i k_i)$ value. Let's take the following example:

$$\begin{aligned} d_1^2 &= (10)_2, \\ d_2^4 &= (0110)_2, \\ d_3^3 &= (110)_2, \end{aligned} \qquad\qquad r = (101|011|10|0)_2$$

Using such a numbering scheme, a data mapping function is defined with the help of following rules:

1. Level $k+1$ is created by splitting each region in a specific dimension.
2. The total number of regions for the level 1 is 2^k.
3. The region $(r, 1)$ can be split into $(r, k+l)$ and $(r + 2^k, k+1)$.
4. Region numbers r and $r + 2^k$ are two possible extensions of the binary representation of r by one bit (the most significant).
5. The value space of a scale is doubled, if the splitting level in dimension i increases from k_i to k_{i+1}. This indicates an extension by one bit.

The value points can now be mapped to a unique region number by performing a simple concatenation of binary representation of the scale values. This efficient mapping is valid if regular

decomposition of the space is cyclic through the set of dimensions. Offsets correspond to different dimensional scales depending on each attribute domain. It is important to mention here that the categorical values are first transformed into numerical values before mapping. We follow the same mechanism as adopted by Iwuchukwu et al. [9]. Mapping is depicted in Algorithm 1.

Algorithm 1: Mapping Scheme [33]

```
Data: r = 0; offset = 1; s = 0
Result: Region numbers
initialization;
while s < k do
    i = s mod n + 1;
    j = s div n + 1;
    if k[i] >= j then
        r = r + offset * b[i, k[i]-j];
        offset = offset * 2;
        s = s + 1;
    end
end
```

3.3. BANG Directory

Despite historical proximity with the Grid file and its variants, the directory of the BANG file is a tree rather than an array (grid). It follows Samet's claim [32], who states that the BANG file is a variant of the *kd*-B-trie, that is, a *kd*-B-tree with regular decomposition. Figure 6 shows an example of a BANG tree directory from partitioning of Figure 5. Blocks are in the leaves, whereas inner nodes contain entries of the form (subspace spanned by a child node, reference to child node). The subspace spanned by a child node is defined as an outermost hyper-rectangle and zero or more nested regions to remove. On Figure 6, we denote by X a simple hyper-rectangle (region) and $X!$ a complex shape built as follows:

$$X! = X - \bigcup Y, \tag{4}$$

where Y is an X-nested region that occurs in the path from the root to the current node. For instance, the root of the BANG tree directory from Figure 6 contains two entries: D and $A!$. D is the sub-space spanned by the left child node, whereas $A! = A - D$ is the sub-space spanned by the right child node. Note that the second $A!$ of the tree is defined as follows:

$$A! = A - (B \cup D). \tag{5}$$

The partitioning algorithm is simple yet efficient. As any $kd - B$-tree, its time complexity is $O(NlogN)$. For external data, I/O cost still remains in $O(\frac{N}{B}.log\frac{N}{B})$ with B the disk block size. It performs incremental insertion of data points in a top-down manner. The enclosing grid region identifier is first computed thanks to the mapping scheme discussed before. The all path up to the root (the entire space) is also retrieved. Then, the BANG directory is searched for the smallest recorded region that encloses the data point. It is then assigned to the corresponding bucket.

When data bucket overflows, the algorithm operates splitting to balance the distribution of points between buckets. When a bucket A overflows, the value space is split and a new bucket B is created, and those records that lie in one half of the space are moved from the old bucket to the new one. Splitting is done by iteratively halving the space spanned by points in the bucket until the best balance is achieved. It gives birth either to a buddy region or to a newly enclosed region. Further details on splitting strategies can be found in [34]. The iterative halving strongly differs from the R^+-tree splitting strategy. Indeed, the BANG file operates from the entire space to blocks (top-down), whereas

the R^+-tree operates from points to blocks (bottom-up). This distinct feature will be discussed further in the Performance section (see Section 5.2).

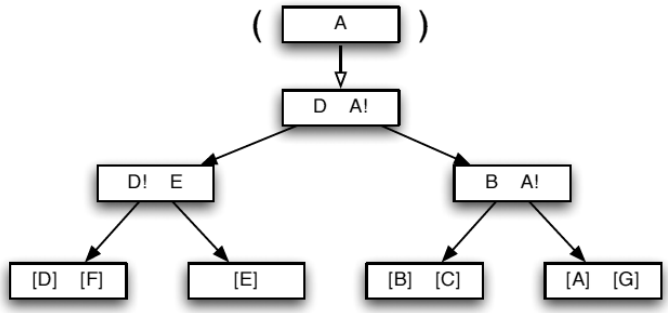

Figure 6. BANG directory of partitioning from Figure 5 represented as a *kd*-B-trie with fanout $B = 2$.

Finally, in order to first encompass anonymity requirements, and then to obtain the highest quality partitioning for the anonymization process, we set up the minimum filling rate to the lowest page size (M) value, depending on the grid resolution. Then, splitting is performed whenever the bucket size reaches $2M + 1$. The second parameter B (the fan-out of the tree directory) is set to the page size such that it allows for building an external tree structure that scales up to potentially any size of data sets.

4. From BANG Directory to Anonymous Public Release

We chose to achieve k-anonymity, based on the BangA generalization algorithm, and this optional post-processing step merges buckets from the BANG file to build equivalence classes of the anonymous release with the desired parameter $k \leq M$. Although the brute BANG directory already achieves very high quality in the public release thanks to non hyper-rectangular blocks, this additional processing increases the usefulness of the anonymous public release for higher k values. Actually, it could be performed on any other axis-parallel partitioning for anonymization and can be performed independently to any PAM algorithm, e.g., the R^+-tree approach.

4.1. Density-Based Clustering

BangA performs a density-based clustering on data buckets in a way similar to BANG-clustering [33]. BANG-clustering is a grid clustering approach that relies on a main memory *kd*-tree accommodated from the BANG file. It is a direct descendant from GRIDCLUS [34], itself based on the Grid file.

The algorithm computes a density index for each block assigned to a data bucket, and it creates a dendrogram by merging neighbor blocks having the closest density indices. However, BANG-clustering as well as GRIDCLUS compute density indices for each block $[X]$ by means of $card([X])$, the number of data points it contains, and $V(X)$, the spatial volume of the enclosing region only. This approach does not leverage the non hyper-rectangular blocks built by the BANG file. Thus, we refine the previous proposal and define the spatial volume $V([X])$ of a BANG directory entry $[X]$ as follows:

$$V([X]) = \prod_{1 \leq i \leq n} e_X^i - \sum_{Y \in \mathcal{H}_X} \prod_{1 \leq i \leq n} e_Y^i, \tag{6}$$

where $X - \bigcup_{\mathcal{H}_X} Y$ is the block where the data points from $[X]$ are located, and X and elements of $\mathcal{H}_X$ are hyper-rectangular regions. The e^i_α are extents of the region α on the ith dimension.
The density index $\mathcal{D}([X])$ of block $[X]$ is then given by the ratio:

$$\mathcal{D}([X]) = \frac{card([X])}{V([X])}. \tag{7}$$

Next, the algorithm performs a sort on blocks according to their decreasing density. The ranking of blocks supports the construction of a *dendrogram* obtained by merging iteratively pairs of *neighbor* blocks with the highest density indices, creating new clusters otherwise. Neighborhood is defined as a shared $(n-1)$-dimensional hyperplane between two block regions. The algorithm is detailed in [34].

4.2. Multi-Granular Anonymity

BangA allows multi-granular anonymity in a single run. However, instead of working directly on the index structure, we leverage the above *dendrogram* by means of computation of a cut. The main purpose is to allow the end-user to set the k value on the fly, without the need for scanning raw data. For the basic $k = M$ setting, leaves of the dendrogram are straightforwardly the equivalence classes for the anonymous public release thanks to the filling requirements on the BANG file. The process to find a required cut in dendrogram is linear in the worst case to the number of nodes because a single path in the tree from the root is a local decision i.e., does the child node of the current node fulfill the cardinality constraint? If yes, we have the desired result; otherwise, the dendrogram traversing continues until cardinality constraints are met. Optimizations such as top-down/bottom-up crossing paths are still possible, but the procedure is already simple and fast (negligible cost compared to the rest).

If we consider higher k values, then we could perform a top-down depth-first traversal of the dendrogram until we reach maximally specialized k- filled blocks in each and every branch. The result cut draws the anonymous public release. Since we compute the cut on the dendrogram rather than on the index structure (*kd-B*-trie), then the k value is not restricted to M^ℓ settings. Indeed, the approach gives BangA the ability to perform cM-anonymity, for any natural number c. In R^+-tree based approach [9], this feature is offered thanks to an ordered leaves scan of the *kd-B*-tree that gives low quality releases compared to BangA since adjacent leaves could be merged even if they belong to very different branches of the tree, which are the bottom line for any clustering technique.

4.3. BangA and Other Syntactic Generalization Models

Though we employed BangA to achieve k-anonymity generalization model, it can be directly exposed to any sophisticated syntactic generalization model e.g., ℓ-diversity [29] and t-closeness [35]. As its R^+-tree counterpart, BangA would be able to incorporate constraints from the definition of the various existing generalization models in its anonymization process. The only accommodation would be to redefine the assignment and splitting strategies such that both resulting blocks satisfy the generalization model. For instance, to make the anonymous release ℓ-diverse, it requires that at least ℓ sensitive values are "well represented" in each equivalence class. Thus, BangA would incorporate checking on sensitive values in its splitting decision to only create new ℓ-diverse blocks from old ones. In addition, it would add constraints on assignment of a new point into an existing block such that the resulting block still satisfies the ℓ-diversity; otherwise, the algorithm would locally redistribute points into blocks. Then, in order to improve the utility of a final ℓ-diverse release, there can be several possible ways of switching the points in the blocks so as to keep the generalization to minimum.

4.4. Compaction Procedure

Iwuchukwu et al. [9] propose a compaction procedure that simply shrinks the envelope of each block to its MBB as shown in Figure 7. The R^+-tree approach natively computes such MBBs for every

block. Consequently, the average volume of the blocks is minimized. However, BangA operates a top-down decomposition of the space such that the union of all the blocks spans the entire space. Obviously, a compaction of each block would yield to a more accurate anonymous public release, and would still increase its quality w.r.t the R^+-tree numbers. Thus, it can be considered as a straightforward improvement of BangA, even if computation of non hyper-rectangular "MBB" , e.g., those on Figure 8 must be carefully defined.

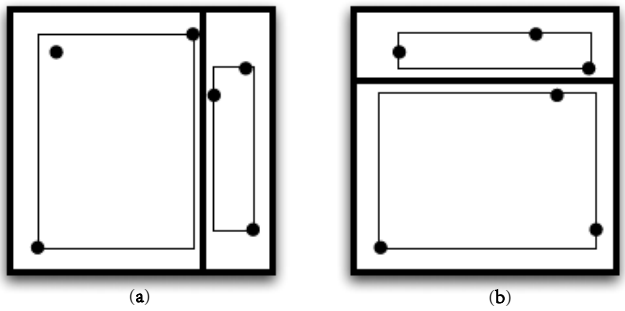

Figure 7. Low quality binary partitioning of a set of six points into blocks of at least three points, following either (**a**) *X*-axis; or (**b**) *Y*-axis.

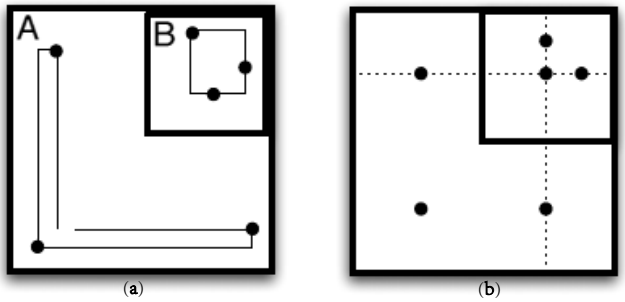

Figure 8. BANG file (NHR, non-hierarchical routing) partitioning with cardinality constraint ($\leq$3 points), (**a**) on points from Figure 7; and (**b**) where HR partitioning fails.

5. Experimental Validation

This section provides an extensive experimentation on BangA generalization algorithm to achieve *k*-anonymous public release.

5.1. Preparation and Settings

We conducted experiments on two data sets for the empirical evaluation of BangA. Efficiency and effectiveness were addressed according to time cost of the computation and quality of the public release, respectively. We also implemented R^+-tree approach for *k*-anonymization espoused in [9] for comparison with BangA, since it is commonly admitted that it is the reference algorithm. Though we had no access to the source code, we implemented the closest possible solution for the given approach strictly observing the requirements in [9] and adopted the same architecture than BangA for the sake of equity in the comparison analysis.

We used the popular "Adults" data set taken from the U.C. Irvine Machine Learning Repository. This data set, also known as "Census Income" data set, contains the data about individuals in the USA. We purged all records with missing values and were left with a table containing 1 million tuples. We used the attributes *Age*, *Zipcode* and *Education level* as quasi-identifiers. The second data set was "Voter list" taken as is from the experiments conducted by Sweeney [1] in her seminal work

on *k*-anonymization. It contains 54,803 records (tuples with missing values are already removed). We used *Age*, *Zipcode* and *Salary* as quasi-identifiers. For stress testing and to study the behavior of both the approaches for high dimensional data, we used a third data set named "Customer" that was synthetically generated using a data generator tool (Datanamic data generator: [36]). This data set contains 1 million tuples with 15 attributes and all of them are used as quasi-identifiers.

To conduct the experiments in a real database environment, we first populated a PostgreSQL database with all three of the data sets. For convenience and code efficiency, data has been normalized (see Section 3.1) on the database level using advanced query facilities provided by PostgreSQL DBMS. We also used database statistics for query optimization. We applied R^+-tree and BangA approaches on all the quasi-identifiers of Adults, Voter list and Customer data sets. In the R^+-tree approach, the anonymization process is followed as in [9]. The algorithms are developed in C++ and are tested on a system with 4GB RAM and a Windows 7 operating system (Microsoft Corporation, Redmond, WA, USA).

5.2. Performance

Execution time of R^+-tree and BangA was measured on Voter list, Adults and Customer data sets—thus from 50 K to 1 Million records. Block size and page size was set to five, in order to evaluate the performance of each algorithm under stress of very small bucket and fan-out values. Many runs with different settings have been done and results always confirm those presented here. In this experiment, we evaluated the BangA algorithm with *dendrogram* construction, against the brute R^+-tree construction, i.e., w/o leaf scan or cut extraction.

Results are presented in Table 2. It shows a 50% lower time cost in favor of BangA compared to R^+-tree for the Voter list data set, and the difference still increases for large data sets like Adults since it spends up to 36% less execution time for the public release computation. For high dimensional data in Customer data set with 15 quasi-identifiers, BangA simply outperforms R^+-tree based anonymization, as the latter is unable to cope with such high dimensional data. In order to make a verifiable comparison of both approaches, a set of 1 million tuples with seven quasi-identifier attributes was randomly sampled from the Customer data set (having 1 million tuples and 15 quasi-identifier attributes). With a slightly large number of quasi-identifiers, results in Table 2 indicate substantially lower time cost in favor of BangA compared to R^+-tree. This shows that whatever the number of dimensions, BangA out performs its counterpart.

Table 2. Time cost (in seconds) of BangA and R^+-tree with $B = 5$ and $M = 5$.

Data Set	Size	Quasi-Identifiers	R^+-Tree	BangA
Voter list	54,803	3	6 s	3 s
Adults	1 million	3	91 s	55 s
Customer	1 million	15	Out of memory	240 s
Customer	1 million	7	1214 s	112 s

We did not compare the efficiency of BangA with previous proposal such as Mondrian [8], since experiments in [9] have shown that R^+-tree anonymization outperforms all of the previous algorithms. Thus, these experiments validate the very good behavior of BangA regarding performance and scalability. Moreover, the second storage structure of the BANG file guarantee that it could handle very large data sets without any drop in the performance.

The second result is as follows: we could argue that bottom-up spatial indexing is not systematically more efficient than top-down approach as conjectured in [9]. This result is given by our own experiments comparing in the same running environment R^+-tree approach (bottom-up) with BangA (top-down). Following the usual analysis on spatial access methods, we claim that the performance is mainly dependent from the splitting strategy. In BangA, we use regular decomposition following the grid, whereas the original R^+-tree grows by means of a quadratic procedure comparing

pairwise distances of elements in an overflowed bucket. These strategies determine a constant factor (w.r.t. N) in time complexity that makes the execution time slower for the R^+-tree as shown on Table 2.

We also empirically evaluated the influence of input parameters on the process. We compared *fine-grained* and *coarse-grained* block sizes both for the R^+-tree and BangA. The results indicate that varying the M parameter, for a given output k value, does not affect the quality of data, but it reduces the execution time of both algorithms, as there would be less partitions in both cases. For instance, to build a 100-anonymous release, it is fast and safe to set $M = 100$ and to build the public release as the set of leaves of the tree directory. However, in this case, the construction of any k-anonymous release, $k < 100$, requires a new run. Thus, for a generic anonymization process, it is much better to set M to a quite small value and to then perform a cut in the the dendrogram given an online k parameter.

5.3. Quality of the Public Release

Since the k-anonymity problem relies on the trade-off between privacy of individuals and utility of the public release, we computed and compared quality of the public releases built, respectively, with BangA and with the R^+-tree, by means of several measures of information loss. The main idea is to evaluate the extent to which the data set has been distorted when generalizing records. We adopted generic quality measures, i.e., measures that do depend on the application domain or on a specific usage of the public release. Then, we first followed the experimental protocol described by Iwuchukwu et al. [9], with three different measures: *discernibility penalty* [3], *KL-divergence* [4] and the certainty metric proposed in [5].

We conducted experiments on the Adults and Customer data sets. Results for Adults data set are presented on Figures 9–11 for certainty metric, discernibility penalty and KL-divergence, respectively. Roughly speaking, all of the experiments show that BangA provides higher quality public releases than the R^+-tree since BangA curves systematically remain lower than those from the R^+-tree and quality measures are actually "penalty" measures.

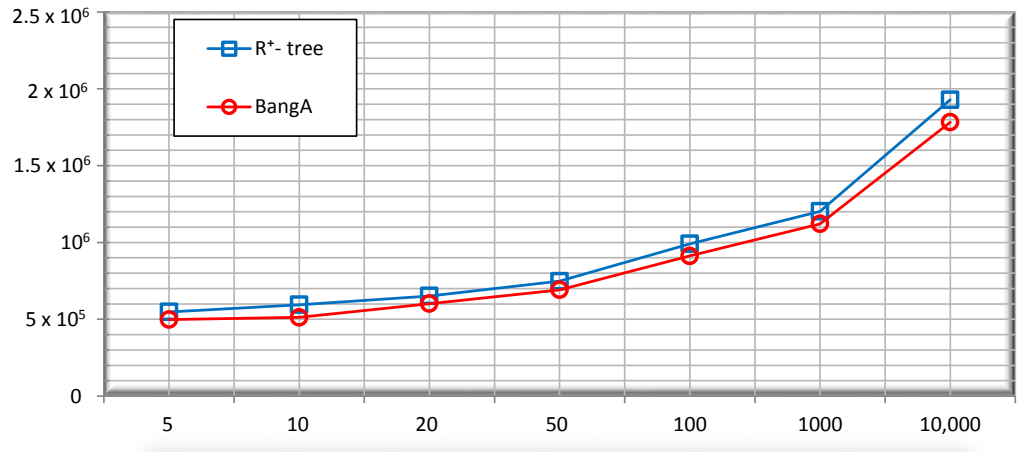

Figure 9. Certainty penalty (Y-axis) according to k parameter (X-axis) in 5, 10, 20, 50, 100, 1000, and 10,000 on a log-linear scale.

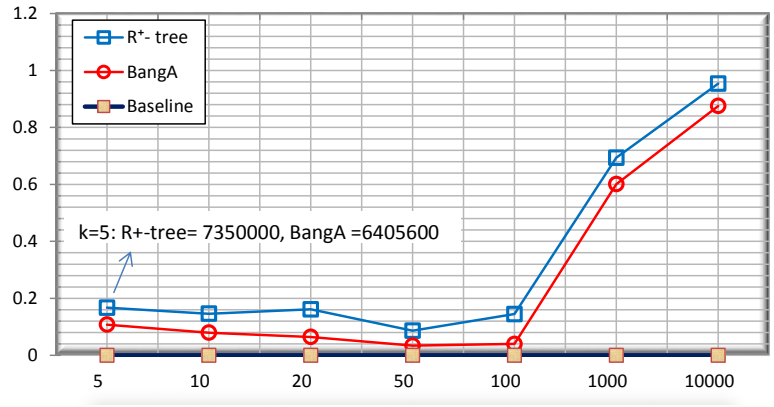

Figure 10. Discernability penalty (on *Y*-axis, normalized by log ratio on baseline) according to *k* parameter (*X*-axis) in 5, 10, 20, 50, 100, 1000, and 10,000.

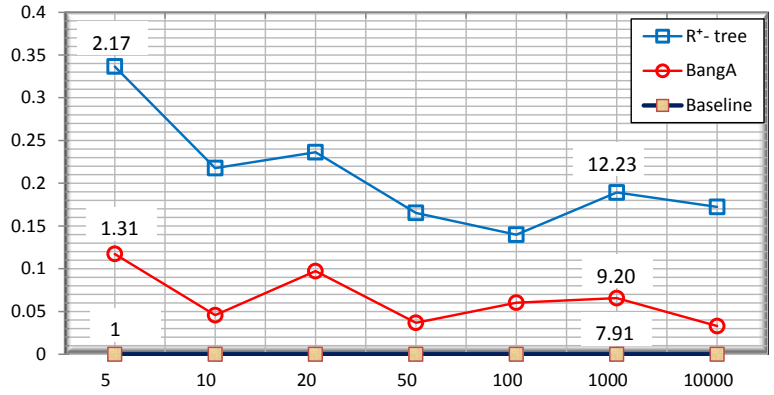

Figure 11. KL-divergence (on *Y*-axis, normalized by log ratio on baseline) according to *k* parameter (*X*-axis) in 5, 10, 20, 50, 100, 1000, and 10,000.

Next, curves are all increasing since the higher the k parameter, the lower the overall quality. We could notice that the gap between the R^+-tree and BangA increases with k in the discernibility penalty. Here, we face the usefulness of the density-based clustering of BangA since merging elementary blocks give birth to very accurate equivalence classes even for higher k values, compared to the R^+-tree. To focus on specific values rather than analysis trends in large scale curves, we consider numbers for $k = 100$ since it represents a descent rate of 0.01% of the size of the data set. Here, we observe 5% better quality in CM, 8% in DP an 9% in KL-divergence always in favor of BangA. For instance, in Figures 10 and 11, we normalized the values, respectively, for KL-divergence and DCP for R^+-tree and BangA w.r.t. baseline values (original values are marked for $k = 5, 1000$ in Figure 11 and for $k = 5$ in Figure 10) in order to highlight the gain achieved by BangA over R^+-tree based anonymization. These values are prototypical of the average gap between BangA and R^+-tree with a varying k value. Moreover, if we consider the baseline of DP, then the improvement of BangA with respect to the R^+-tree is more than 48%. Finally, it is worth noticing that CM is not designed to take into account non hyper-rectangular blocks since it aggregates dimensional range values. Thus, we only computed estimated values based on enclosing regions for BangA.

5.4. Query Accuracy

Apart from studying the quality of data through "penalty" measures and KL-divergence metrics, the utility of the anonymized data is also studied in terms of *relative query error*. In this section, we focus

on *point and window queries* as they are important building blocks for statistical analysis and many data mining applications (e.g., association rule mining and decision trees).

This approach aims at measuring the utility of a sanitized release in terms of accuracy in answering aggregate queries. For answering the aggregate queries, the "COUNT" operator is considered in which the query predicate includes quasi-identifier attributes. Let R be a table with q quasi-identifiers, $QI_1, \ldots, QI_q$, where $D(QI_i)$ denotes the domain of i_{th} quasi-identifier. Then, the queries are of the form:

SELECT COUNT$(*)$ **from** R
WHERE $qi_1 \in D(QI_1)$
AND ... **AND** $qi_q \in D(QI_q)$

The predicate of a query contains two important parameters: (1) the query dimensionality parameter q; and (2) the query selectivity θ. The query dimensionality parameter q indicates the number of quasi-identifiers used in the predicate. The query selectivity θ indicates the number of values for each attribute A_j, $1 \leq j \leq n$. Query selectivity is usually obtained as follows:

$$\theta = \frac{|T_Q|}{|R|}, \tag{8}$$

where $|T_Q|$ is the number of tuples in the result set obtained from Q on R and $|R|$ is the number of tuples in data set. The error for the query Q, denoted Error(Q), is the normalized difference between the result set from the evaluation of Q on raw and sanitized data, respectively. Then, the query error is calculated as follows:

$$\text{Error}(Q) = \frac{\text{sanitized\_count} - \text{actual\_count}}{\text{actual\_count}}, \tag{9}$$

where the result from the COUNT query on R is denoted by *actual_count* and on R^* as *sanitized_count*.

We used the randomly sampled Customer data set containing 1 million tuples and seven quasi-identifier attributes for these experiments.

The point queries are relatively easier to handle. The window queries are of the form:

SELECT COUNT$(*)$ **from** R
WHERE $R.QI_1 \geq qi_1$ **AND** $R.QI_1 \leq qi_2$
AND
...
AND
$R.QI_7 \geq qi_7$ **AND** $R.QI_7 \leq qi_7$

The above mentioned seven-dimensional query is dynamically created by using the upper and lower bounds on the range of each participating attribute. These bounds are defined as follows:
A *COUNT* query Q on the anonymized data set R^* fetches the count of tuples matching the query Q. For point query, the result set contains those tuples with the lowest value on the region level.

A window query Q returns a count of the records in R^* that matches Q. A tuple $t \in R^*$ is said to be a matching tuple for Q if the region spanned by t and the query Q have a non-null intersection i.e., t must intersect Q on all quasi-identifier attributes.

We conducted the experiments using query dimensionality parameter. The query error rate is calculated using Equation (9). We considered 300 randomly generated queries for conducting these experiments and calculated the average relative error.

For these experiments, we anonymized the Customer data set on all seven quasi-identifiers and varied the query dimensionality parameter i.e., the number of QI attributes in query predicate. The results for point and window queries with varying query dimensionality are shown in Figures 12 and 13. As the query dimension increases, average relative error rate decreases. Thus, the anonymized data performs better for queries with a larger query dimensions. BangA tends to be more stable than an R^+ based approach showing less relative error rate for any query dimension.

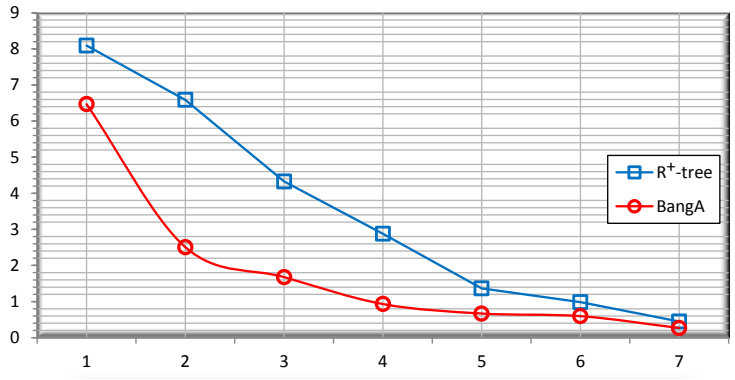

Figure 12. Error (*Y*-axis) for point queries according to varying query dimensionality (*X*-axis).

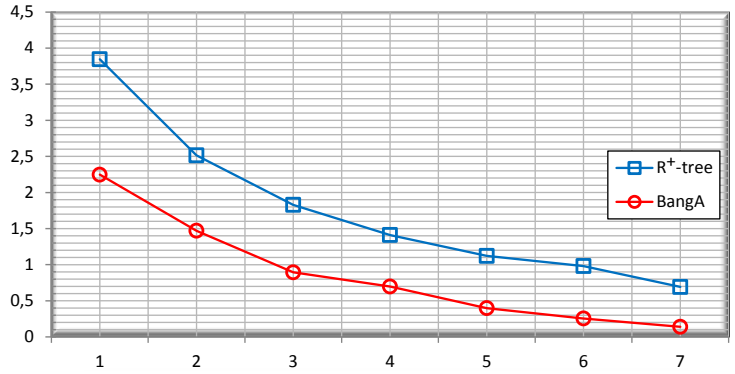

Figure 13. Error (*Y*-axis) for window queries according to varying query dimensionality (*X*-axis) with dimensions = 7.

6. Conclusions

In this communication, we proposed a new generalization algorithm called BangA. Based on the BANG file indexing structure, it performs very well and provides non hyper-rectangular blocks assigned to the equivalence classes of the public release. Furthermore, BangA allows for incorporating background knowledge in the dimensional scales that are used for regular decomposition. A post-processing step provides a density-based clustering of the blocks in order to achieve a high quality anonymization regardless of the k value, and, since the result of such post-processing is a dendrogram, then it offers the opportunity to build on demand k-anonymous release without reconstructing the dendrogram again. Along with the usual benefits, BangA can easily be extended to adopt the compaction procedure to achieve better utility of data. In addition, BangA can incorporate other generalization models like ℓ-diversity by making slight adjustments in its assignment and splitting strategies.

As described in Section 4.3, BangA can be directly applied to any syntactic generalization model. Quite recently, DP has emerged as a state-of-the-art semantic privacy paradigm that offers strong theoretical privacy guarantees. Due to its inability of achieving practical implementation, there is a surge of works nowadays that tend to combine the practicalness of syntactic approaches with the effectiveness of DP. Since BangA performs very efficient multidimensional partitioning to achieve high quality generalization, inspired by Gehrke et al. [26] DP relaxation, as a future work, we may adopt the following framework for achieving DP:

- random sampling,
- regular generalization mechanism with multidimensional partitioning,
- Laplace mechanism for blocks with small counts.

BangA has been proved to manage sanitization with a generalization mechanism in a very efficient way and with a high quality output. Then, we legitimately propose to settle a sanitization process to achieve crowd blending with BangA.

Acknowledgments: This research is supported by HEC Pakistan and Laboratoire LINA, University of Nantes.

Author Contributions: Adeel Anjum conducted proof of concept experiments and contributed to the overall writing of the article, Guillaume Raschia wrote the main content of the article and completed the proofreading. Both authors have read and approved the final manuscript.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Sweeney, L. Achieving *k*-anonymity privacy protection using generalization and suppression. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.* **2002**, *10*, 571–588.
2. Agarwal, P. Range searching. In *Handbook of Discrete and Computational Geometry*; CRC Press, Inc.: Boca Raton, FL, USA, 1997; Volume 2, pp. 809–838.
3. Bayardo, R.; Agrawal, R. Data privacy through optimal *k*-anonymization. In Proceedings of the 21st International Conference on Data Engineering (ICDE 2005), Tokyo, Japan, 5–8 April 2005; pp. 217–228.
4. Kifer, D.; Gehrke, J. Injecting utility into anonymized datasets. In Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data, Chicago, IL, USA, 27–29 June 2006; ACM: New York, NY, USA, 2006; p. 228.
5. Xu, J.; Wang, W.; Pei, J.; Wang, X.; Shi, B.; Fu, A. Utility-based anonymization using local recoding. In Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Philadelphia, PA, USA, 20–23 August 2006; ACM: New York, NY, USA, 2006; p. 790.
6. Meyerson, A.; Williams, R. On the complexity of optimal *k*-anonymity. In Proceedings of the Twenty-Third ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, Paris, France, 14–16 June 2004; ACM: New York, NY, USA, 2004; pp. 223–228.
7. Fung, B.; Wang, K.; Chen, R.; Yu, P.S. Privacy-preserving data publishing: A survey of recent developments. *ACM Comput. Surv. (CSUR)* **2010**, *42*, 14.
8. LeFevre, K.; DeWitt, D.; Ramakrishnan, R. Mondrian multidimensional *k*-anonymity. In Proceedings of the 22nd International Conference on Data Engineering ICDE'06, Atlanta, GA, USA, 3–7 April 2006; p. 25.
9. Iwuchukwu, T.; Naughton, J. *K*-anonymization as spatial indexing: Toward scalable and incremental anonymization. In Proceedings of the 33rd International Conference on Very Large Data Bases, VLDB Endowment, Vienna, Austria, 23–27 September 2007; pp. 746–757.
10. Ghinita, G.; Kalnis, P.; Khoshgozaran, A.; Shahabi, C.; Tan, K. Private queries in location based services: Anonymizers are not necessary. In Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, Vancouver, BC, Canada , 9–12 June 2008; ACM: New York, NY, USA, 2008; pp. 121–132.
11. Gruteser, M.; Grunwald, D. Anonymous usage of location-based services through spatial and temporal cloaking. In Proceedings of the 1st International Conference on Mobile Systems, Applications and Services, San Francisco, CA, USA, 5–8 May 2003; pp. 31–42.
12. Damiani, M.; Bertino, E.; Silvestri, C. The probe framework for the personalized cloaking of private locations. *Trans. Data Priv.* **2010**, *3*, 123–148.
13. Mokbel, M.; Chow, C.; Aref, W. The new casper: Query processing for location services without compromising privacy. In Proceedings of the 32nd International Conference on Very Large Data Bases, Seoul, Korea, 12–15 September 2006; pp. 763–774.
14. Chor, B.; Goldreich, O.; Kushilevitz, E.; Sudan, M. Private information retrieval, In Proceedings of the 36th Annual Symposium on Foundations of Computer Science, Milwaukee, WI, USA, 23–25 October 1995; pp. 41–50.
15. Krishnamachari, B.; Ghinita, G.; Kalnis, P. Privacy-preserving publication of user locations in the proximity of sensitive sites. In *Scientific and Statistical Database Management*; Springer: New York, NY, USA, 2008; pp. 95–113.
16. Navarro-Arribas, G.; Torra, V.; Erola, A.; Castellà-Roca, J. User *k*-anonymity for privacy preserving data mining of query logs. *Inf. Process. Manag.* **2012**, *48*, 476–487.

17. Erola, A.; Castellà-Roca, J.; Navarro-Arribas, G.; Torra, V. Semantic microaggregation for the anonymization of query logs. In *Privacy in Statistical Databases*; Springer: Berlin/Heidelberg, Germany, 2010; pp. 127–137.
18. Aggarwal, G.; Feder, T.; Kenthapadi, K.; Khuller, S.; Panigrahy, R.; Thomas, D.; Zhu, A. Achieving anonymity via clustering. In Proceedings of the Twenty-Fifth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, Chicago, IL, USA, 26–28 June 2006; ACM: New York, NY, USA, 2006; pp. 153–162.
19. Byun, J.; Kamra, A.; Bertino, E.; Li, N. Efficient *k*-anonymization using clustering techniques. In Proceedings of the Advances in Databases: Concepts, Systems and Applications, Bangkok, Thailand, 9–12 April 2007; pp. 188–200.
20. Chiu, C.; Tsai, C. A *k*-anonymity clustering method for effective data privacy preservation. In Proceedings of the Advanced Data Mining and Applications, Harbin, China, 6–8 August 2007; pp. 89–99.
21. Pilevar, A.; Sukumar, M. Gchl: A grid-clustering algorithm for high-dimensional very large spatial data bases. *Pattern Recognit. Lett.* **2005**, *26*, 999–1010.
22. Wang, W.; Yang, J.; Muntz, R. STING: A statistical information grid approach to spatial data mining. In Proceedings of the International Conference on Very Large Data Bases, Athens, Greece, 26–29 August 1997; pp. 186–195.
23. Samarati, P.; Sweeney, L. Generalizing data to provide anonymity when disclosing information. In Proceedings of the ACM SIGACT SIGMOD SIGART Symposium on Principles of Database Systems, Seattle, WA, USA, 1–3 June 1998; Volume 17, p. 188.
24. Aggarwal, C. On *k*-anonymity and the curse of dimensionality. In Proceedings of the 31st international conference on Very Large Data Bases, VLDB Endowment, Trondheim, Norway, 30 August–2 September 2005; pp. 901–909.
25. Li, N.; Qardaji, W.; Su, D. Provably Private Data Anonymization: Or, k-Anonymity Meets Differential Privacy. Available online: https://www.cerias.purdue.edu/assets/pdf/bibtex_archive/2010-24.pdf (accessed on 29 December 2016).
26. Gehrke, J.; Hay, M.; Lui, E.; Pass, R. Crowd-blending privacy. In *Advances in Cryptology–CRYPTO 2012*; Springer: Berlin/Heidelberg, Germany, 2012; pp. 479–496.
27. Soria-Comas, J.; Domingo-Ferrer, J.; Sánchez, D.; Martínez, S. Enhancing data utility in differential privacy via microaggregation-based *k*-anonymity. *VLDB J.* **2014**, *23*, 771–794.
28. Sweeney, L. *k*-anonymity: A model for protecting privacy. *Int. J. Uncertain. Fuzziness Knowl. Based Syst.* **2002**, *10*, 557–570.
29. Machanavajjhala, A.; Kifer, D.; Gehrke, J.; Venkitasubramaniam, M. l-diversity: Privacy beyond *k*-anonymity. *ACM Trans. Knowl. Discov. Data (TKDD)* **2007**, *1*, 3.
30. LeFevre, K.; DeWitt, D.J.; Ramakrishnan, R. Incognito: Efficient full-domain *k*-anonymity. In Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data, Baltimore, MD, USA, 14–16 June 2005; ACM: New York, NY, USA, 2005; p. 60.
31. Freeston, M. The BANG file: A new kind of grid file. *ACM SIGMOD Rec.* **1987**, *16*, 260–269.
32. Samet, H. *Foundations of Multidimensional and Metric Data Structures*; Morgan Kaufmann: Burlington, MA, USA, 2006.
33. Schikuta, E.; Erhart, M. The BANG-clustering system: Grid-based data analysis. In *Advances in Intelligent Data Analysis Reasoning about Data*; Springer: Berlin/Heidelberg, Germany, 1997; pp. 513–524.
34. Schikuta, E. Grid-clustering: An efficient hierarchical clustering method for very large data sets. In Proceedings of the 13th International Conference on Pattern Recognition, Vienna, Austria, 25–29 August 1996; Volume 2, pp. 101–105.
35. Li, N.; Li, T.; Venkatasubramanian, S. *t*-closeness: Privacy beyond *k*-anonymity and *l*-diversity. In Proceedings of the IEEE 23rd International Conference on Data Engineering, ICDE 2007, Istanbul, Turkey, 16–20 April 2007; pp. 106–115.
36. Datanamic Data Generator. Available online: http://www.datanamic.com/datagenerator/index.html (accessed on 31 December 2016).