Volume 3 • Issue 2 • April 2012

INTERNATIONAL JOURNAL OF DATA ENGINEERING (IJDE)

ISSN : 2180-1274
Publication Frequency: 6 Issues / Year

CSC PUBLISHERS
http://www.cscjournals.org

INTERNATIONAL JOURNAL OF DATA ENGINEERING (IJDE)

VOLUME 3, ISSUE 2, 2012

EDITED BY
DR. NABEEL TAHIR

ISSN (Online): 2180-1274

International Journal of Data Engineering is published both in traditional paper form and in Internet. This journal is published at the website http://www.cscjournals.org, maintained by Computer Science Journals (CSC Journals), Malaysia.

IJDE Journal is a part of CSC Publishers

Computer Science Journals

http://www.cscjournals.org

INTERNATIONAL JOURNAL OF DATA ENGINEERING (IJDE)

Book: Volume 3, Issue 2, April 2012
Publishing Date: 16-04-2012
ISSN (Online): 2180-1274

IJDE Journal is a part of CSC Publishers
http://www.cscjournals.org

Published in Malaysia

Typesetting: Camera-ready by author, data conversation by CSC Publishing Services – CSC Journals, Malaysia

CSC Publishers, 2012

EDITORIAL PREFACE

This is second issue of volume three of the International Journal of Data Engineering (IJDE). IJDE is an International refereed journal for publication of current research in Data Engineering technologies. IJDE publishes research papers dealing primarily with the technological aspects of Data Engineering in new and emerging technologies. Publications of IJDE are beneficial for researchers, academics, scholars, advanced students, practitioners, and those seeking an update on current experience, state of the art research theories and future prospects in relation to computer science in general but specific to computer security studies. Some important topics cover by IJDE is Annotation and Data Curation, Data Engineering, Data Mining and Knowledge Discovery, Query Processing in Databases and Semantic Web etc.

The initial efforts helped to shape the editorial policy and to sharpen the focus of the journal. Starting with volume 3, 2012, IJDE appears in more focused issues. Besides normal publications, IJDE intend to organized special issues on more focused topics. Each special issue will have a designated editor (editors) – either member of the editorial board or another recognized specialist in the respective field.

This journal publishes new dissertations and state of the art research to target its readership that not only includes researchers, industrialists and scientist but also advanced students and practitioners. The aim of IJDE is to publish research which is not only technically proficient, but contains innovation or information for our international readers. In order to position IJDE as one of the top International journal in Data Engineering, a group of highly valuable and senior International scholars are serving its Editorial Board who ensures that each issue must publish qualitative research articles from International research communities relevant to Data Engineering fields.

IJDE editors understand that how much it is important for authors and researchers to have their work published with a minimum delay after submission of their papers. They also strongly believe that the direct communication between the editors and authors are important for the welfare, quality and wellbeing of the Journal and its readers. Therefore, all activities from paper submission to paper publication are controlled through electronic systems that include electronic submission, editorial panel and review system that ensures rapid decision with least delays in the publication processes.

To build its international reputation, we are disseminating the publication information through Google Books, Google Scholar, Directory of Open Access Journals (DOAJ), Open J Gate, ScientificCommons, Docstoc and many more. Our International Editors are working on establishing ISI listing and a good impact factor for IJDE. We would like to remind you that the success of our journal depends directly on the number of quality articles submitted for review. Accordingly, we would like to request your participation by submitting quality manuscripts for review and encouraging your colleagues to submit quality manuscripts for review. One of the great benefits we can provide to our prospective authors is the mentoring nature of our review process. IJDE provides authors with high quality, helpful reviews that are shaped to assist authors in improving their manuscripts..

Editorial Board Members
International Journal of Data Engineering (IJDE)

EDITORIAL BOARD

TABLE OF CONTENTS

Volume 3, Issue 2, April 2012

Pages

Clustering Using Shared Reference Points Algorithm Based On a Sound Data Model

Mohamed A. Abbas *mohamed.alyabbas@gmail.com*
Graduate student with the College of Computing and Information Technology
Arab Academy for Science and Technology
Alexandria, P.O. Box 1029, *Egyt.*

Amin A. Shoukry *amin.shoukry@ejust.edu.eg*
Department of Computer Science and Engineering
Egypt-Japan University of Science and Technology
New Borg El-Arab City, P.O. Box 179, Egypt

Abstract

A novel clustering algorithm CSHARP is presented for the purpose of finding clusters of arbitrary shapes and arbitrary densities in high dimensional feature spaces. It can be considered as a variation of the Shared Nearest Neighbor algorithm (SNN), in which each sample data point votes for the points in its k-nearest neighborhood. Sets of points sharing a common mutual nearest neighbor are considered as dense regions/ blocks. These blocks are the seeds from which clusters may grow up. Therefore, CSHARP is not a point-to-point clustering algorithm. Rather, it is a block-to-block clustering technique. Much of its advantages come from these facts: Noise points and outliers correspond to blocks of small sizes, and homogeneous blocks highly overlap. The proposed technique is less likely to merge clusters of different densities or different homogeneity. The algorithm has been applied to a variety of low and high dimensional data sets with superior results over existing techniques such as DBScan, K-means, Chameleon, Mitosis and Spectral Clustering. The quality of its results as well as its time complexity, rank it at the front of these techniques.

Keywords: Shared Nearest Neighbors, Mutual Neighbors, Spatial Data, High Dimensional Data, Time Series, Cluster Validation.

1. INTRODUCTION

The present paper is a modified version of [25]. The modification includes the incorporation of a new measure of cluster homogeneity (section 2.2) which has been used in defining a strict order for cluster's propagation in the proposed algorithm (section 3). Also, new validation indexes; as well as new data sets; have been adopted for the purpose of comparing the proposed algorithm with the previous techniques (section 4).

Clustering of data is an important step in data analysis. The main goal of clustering is to divide data objects into well separated groups so that objects lying in the same group are more similar to one another than to objects in other groups.

Given a set $\mathbf{P} = \{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_N\}$ of data objects (sample points) to be clustered, where $\mathbf{p}_i$ is the i-th object. $\mathbf{p}_i$ is an n-dimensional column vector. $\mathbf{p}_i = [\mathbf{p}_{i1}, \mathbf{p}_{i2}, \ldots, \mathbf{p}_{in}]^T$ consisting of n measured attributes (n is the dimensionality of the feature space). The Jarvis-Patrick clustering technique [12], needs a measure of the distance between two objects and two integers: K and L. K is the size of the neighborhood list, and L is the number of common neighbors. This method works as follows:
Determine the K-nearest neighbors for each object in the set to be clustered. Two objects are placed in the same cluster if they are contained in each other's K nearest neighbors list: $p_i \in$ K-NB$_j$ and $p_j \in$ K-NB$_i$, where K-NB$_j$ and K-NB$_i$ denote the K-nearest neighbors of data points p_j and p_i, respectively. (1) They have at least L nearest neighbors in common: $|NB_j \cap NB_i| \geq L$. (2)

As stated in [15], the principle of K-NB consistency of a cluster states that for any data object in a cluster its K-Nearest Neighbors should also be in the same cluster. The principle of K-Mutual Nearest-Neighbor consistency (K-MNB consistency) states that for any data object in a cluster its K-Mutual Nearest-Neighbors should also be in the same cluster. The principle of cluster K-MNB consistency is stronger than the cluster K-NB consistency concept, and it is also a more strict representation of the natural grouping in the definition of clustering. In the present work, the concept of K-MNB is used in developing a new clustering algorithm, CSHARP, (Clustering using SHAred Reference Points). CSHARP is a density based clustering algorithm in which dense regions are identified using mutual nearest neighborhood. Then, sets of points sharing a common mutual nearest neighbor are considered instead of points themselves in an agglomerative process.

1.1 Proposed Technique

Specifically, the technique proposed in this paper:

- Determines; for every data point; the largest possible set of points satisfying condition (1) only (the cardinality of this set lies in the range $[1 \dots K]$). For a point "p", this set is called its Reference-List and is denoted RL$_{p_i}$. Point "p" is considered as the "Representative Point" (or "Reference Point") of this Reference-List. Such sets of points are the seeds from which clusters may grow up.
- Avoids an early commitment to condition (2), and, instead, proceeds directly from point(s) to set(s) relation(s) (the Reference-Lists); instead of point(s) to point(s) relation(s) (as required by conditions (1) and (2), above). Therefore, data in CSHARP is processed as blocks of points not as individual points. Reference lists constitute a key concept in the proposed algorithm. To preserve K-MNB consistency, CSHARP processes data as blocks of tiny groups of Reference-Lists, on which clusters are built, agglomeratively.
- Allows clusters to be linked if their Reference- Lists share a number of points $\geq M$. Thus, parameter M controls the extent to which mutual neighborhood consistency is satisfied. Allowing clusters to grow in a chain is similar; though not identical; to the reachability relation in DBScan algorithm [5]. Reference-Lists of size $< T$ (a selected threshold) are either considered as noise or may merge with other clusters as will be explained in section 2.

Several experiments conducted on a variety of data sets show the efficiency of the proposed technique for the detection of clusters of different sizes, shapes and densities; whether in low or high dimensional feature spaces; in the presence of noise and outliers. Although the motivations behind the algorithm are quite heuristic, the experimental work showed the validity of the proposed approach.

1.2 Overview of Related Algorithms

Many clustering techniques have been developed based on different concepts. Several approaches utilize the concept of cluster center or centroid, other methods build clusters based on the density of the objects, and a lot of methods represent the data objects as vertices of graphs where edges represent the similarity between these objects.

Centroid based algorithms represent each cluster by using the centre of gravity of its instances. The most well-known centroid algorithm is the K-means [11]. The K-means method partitions a data set into k subsets such that all points in a given subset are close to the same centre. K-means then computes the new centers by taking the mean of all data points belonging to each cluster. The operation is iterated until there is no change in the centers locations. The result strongly depends on the initial guess of centroids, besides, it does not perform well on data with outliers or with clusters of different sizes or non globular shapes.

The key idea of density-based clustering is that for each instance of a cluster, a neighborhood of a given radius has to contain at least a minimum number of instances. One of the most well known density-based clustering algorithms is the DBScan [5]. In DBScan the density associated with an object is obtained by counting the number of objects in a region of a specified radius, ε, around the object. An object with density greater than or equal to a specified threshold, MinPts, is treated as a core (dense), otherwise it is considered as a non-

core (sparse) object. Non-core objects that do not have a core object within the specified radius are discarded as noise. Clusters are formed around core objects by finding sets of density connected objects that are maximal with respect to density-reachability. While DBScan can find clusters of arbitrary sizes and shapes, it cannot handle data containing clusters of different densities, since it uses global density parameters, MinPts and ε, which specify only the lowest possible density of any cluster.

Chameleon [13] and SNN [4] algorithms attempt to obtain clusters with variable sizes, shapes and densities based on K-nearest neighbor graphs. Chameleon finds the clusters in a data set by using a two-phase algorithm. In the first phase, it generates a K-nearest neighbor graph that contains links between a point and its K-nearest neighbors. Then it uses a graph partitioning algorithm to cluster the data items into a large number of relatively small sub-clusters. During the second phase, it uses an agglomerative hierarchical clustering algorithm to find the genuine clusters by repeatedly combining together these sub-clusters. The Shared Nearest Neighbors clustering algorithm, SNN [4] uses K-nearest neighbor approach for density estimation. It constructs a K-nearest neighbor graph in which each data object corresponds to a node which is connected to the nodes corresponding to its K-nearest neighbors. From the K-nearest neighbor graph a shared nearest neighbor graph is constructed, in which edges exist only between data objects that have each other in their nearest neighbor lists. A weight is assigned to each edge based on the number and ordering of shared neighbors. Clusters are obtained by removing all edges from the shared nearest neighbor graph that have a weight below a certain threshold t.

A recent clustering algorithm, Mitosis [22], is proposed for finding clusters of arbitrary shapes and arbitrary densities in high dimensional data. Unlike previous algorithms, it uses a dynamic model that combines both local and global distance measures. The model is depicted in the proposed dynamic range neighborhood, and the proposed clustering criteria which use distance relatedness to merge patterns together. Mitosis uses two main parameters f and k. Parameter f, controlling the neighborhood size, is the main parameter which decides the lower bound on the number of clusters that can be obtained. The value of f, should be varied in an incremental fashion so as not to deteriorate the speed of the algorithm. It can be selected just above the value of 1, and increased by small steps, to avoid unnecessary large neighborhood sizes. Parameter k controls the degree of merging patterns/clusters together, within the limits of the neighborhood decided by f. Increasing values of k, for the same f value, means decreasing the number of clusters obtained, and vice versa.

In recent years, spectral clustering [20] has become one of the most popular modern clustering techniques. It starts from a similarity matrix between data objects, modifies it to a sparse matrix, then computes its Laplacian matrix "L". The first k eigenvectors of L constitute the first k columns of a matrix V. K-means algorithm is applied then on the row vectors of a normalized version of matrix V (where each original data object is assigned to the same cluster to which the corresponding row vector in V is assigned to). Spectral clustering is simple to implement and can be solved efficiently on standard linear algebra software. Additionally, it is more effective in finding clusters than some traditional algorithms such as K-means. However, it suffers from a scalability problem discussed in [17]. It cannot successfully cluster data sets that contain structures at different scales of size and density. To overcome these limitations, a novel spectral clustering algorithm [2] is proposed for computing spectral clustering using a sparse similarity matrix.

1.3 Outline of the Paper

The rest of this paper is organized as follows. Section 2 describes our approach for the definition of similarity and density (or reference points structure), which is the key concept to our clustering algorithm. Section 3 describes the algorithm itself followed by a logical model of the data entities it manipulates, a graph-based interpretation of its effect and its time complexity analysis. An anatomy of the proposed algorithm with respect to related algorithms is discussed next. Section 4 presents the data sets used in the experiments conducted to evaluate the performance of the algorithm using well known cluster validation indexes. Section 5 presents a short conclusion and possible future work.

2. BASIC DEFINITIONS

In this section we describe the basic concepts used in the proposed algorithm.

2.1 Reference-List and Reference-Block

As shown in Figure 1, although the Euclidean distance is a symmetric metric, from the SNN perspective (and considering K=4), point A is in 4-NB_B, however, point B is not in 4-NB_A. Given a set of points P, let $P = \{p_1, p_2, \ldots, p_k, p_{k+1}, p_{k+2}, \ldots, p_N\}$ be the ordered list of points according to their distances from a point p_i, and let K-$NB_{p_i} = \{p_{i1}, p_{i2}, p_{i3}, \ldots, p_{ik}\}$ be the K-nearest neighbors of p_i. This represents the list of points that point p_i refers to. If $p_j \in$ K-NB_{p_i}, then p_j is referred by p_i. Let RB_{p_i} denote the set of points having p_i in their K-Nearest Neighborhood. Then, K-$NB_{p_i} \cap RB_{p_i}$ represents a set of dense points called a Reference-List, RL_{p_i}, associated with point p_i. p_i is called the representative point of RL_{p_i}. Each representative point p_i with its Reference-List, RL_{p_i} constitute a Reference-Block. Points in a Reference-Block are shown in Figure 2.

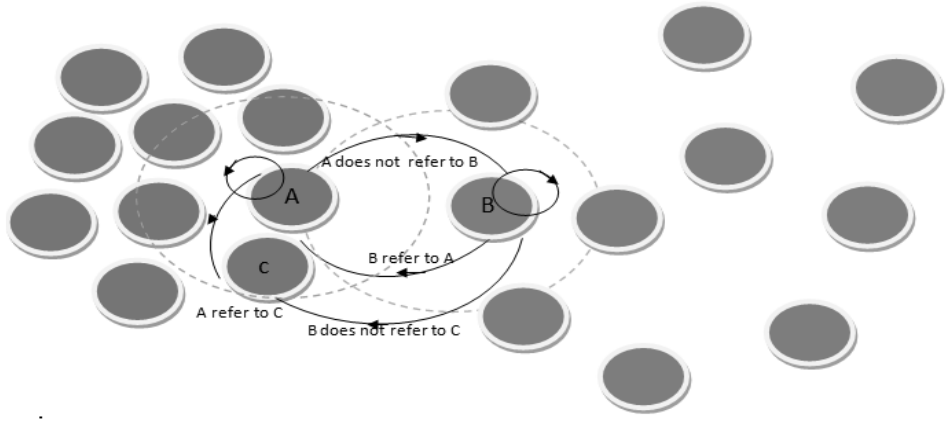


FIGURE 1: Concept of mutual neighboring: "A" is in 4-NBB, however, "B" is not in 4-NBA.

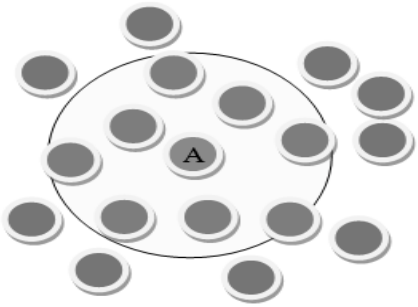


FIGURE 2: A Reference-List (points within circle), associated with a representative point "A"). All together they constitute a Reference-block.

There are three possible types of representative points:

- Strong Points (or Reference Points), representing blocks of size greater than a pre-defined threshold parameter T.
- Noise points, representing empty Reference-Lists. These points will be excluded initially from final clusters.
- Weak points which are neither strong points nor noise points. These points may be merged with another existing clusters if they are members of another strong points.

2.2 Homogeneity Factor

A homogeneity measure is proposed here for the ordering of the cluster's propagation process. While CSHARP[25] performs this process according to the cardinality of the reference lists (i.e their densities), the algorithm proposed here, considers both the density and homogeneity of the blocks that will be granted priority to propagate first.

Given $RLp_i = \{q_1, q_2, \ldots q_c\}$, a Reference List associated with point p_i having cardinality $c = |RLp_i| > T$ (i.e. corresponding to a strong reference point), its homogeneity factor α_i is computed as:

$$\alpha_i = \frac{Avg_{p_{ij}}}{Max_{p_{ij}}} \tag{3}$$

Where $Avg_{p_{ij}}$ is the average distance between p_i and its associated reference list points q_j, computed as:

$$Avg_{p_{ij}} = \frac{1}{c}\sum_{j=1}^{c} |p_i - q_j|,\ q_j \in RLp_i \tag{4}$$

and $Max_{p_{ij}}$ is the maximum distance between p_i and its associated reference list points q_j, computed as:

$$\max |p_i - q_j|,\ q_j \in RLp_i \tag{5}$$

Figure 3 shows three cases of strong reference lists with their corresponding homogeneity factors. Note that the odd case of a reference list with cardinality c=1 in which α_i=1 is excluded, since it does not correspond to a strong reference list. In the experiments described in section four, T is always greater than 2.

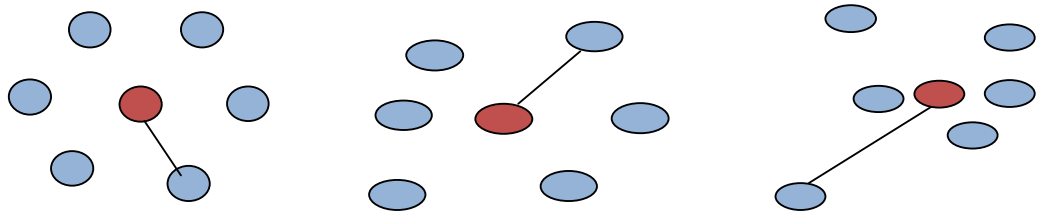


FIGURE 3: Three reference lists with different homogeneity factors (a) 0.98, (b) 0.80 and (c) 0.55

2.3 Cluster's Propagation

Given two clusters c_i, c_j such that $|c_i \cap c_j| \geq M$; where M is a chosen parameter, called merge-parameter, then the two clusters can be merged together. Hence, M measures the minimum link strength required between two clusters. In CSHARP data is processed as blocks of points (reference-blocks) not as individual points. Two clusters are merged if they share a number of points equal to or greater than some threshold M; as explained in Figure 4.

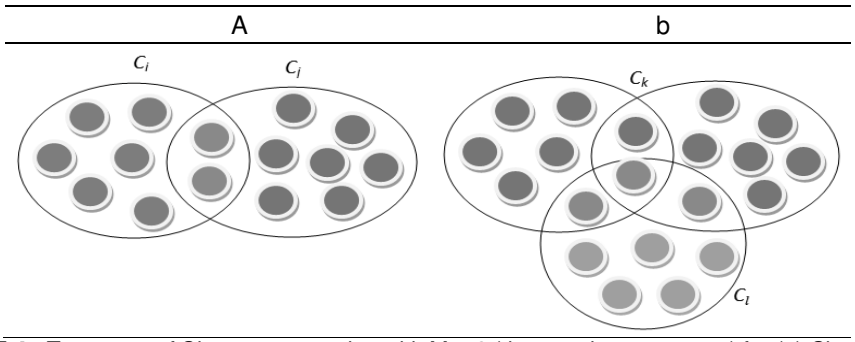


FIGURE 4: Two steps of Cluster propagation with M = 2 (the merging parameter) for (a) Ck = $|c_i \cup c_j|$ such that $|c_i \cap c_j| \geq$ M and (b) Cu = Ck ∪ Cl such that $|c_k \cap c_l| \geq$ M.

To illustrate the process of clusters propagation, Chameleon's data set DS5 is used. DS5 consists of 8000 spatial data points. All genuine clusters were detected at the parameters setting(K=24, T=18, M=6). The number of strong points obtained were 5122.

Several snapshots of the clustering process are shown in Figure 5. Figure 5, also, illustrates how clusters propagate agglomeratively, and simultaneously, in CSHARP.

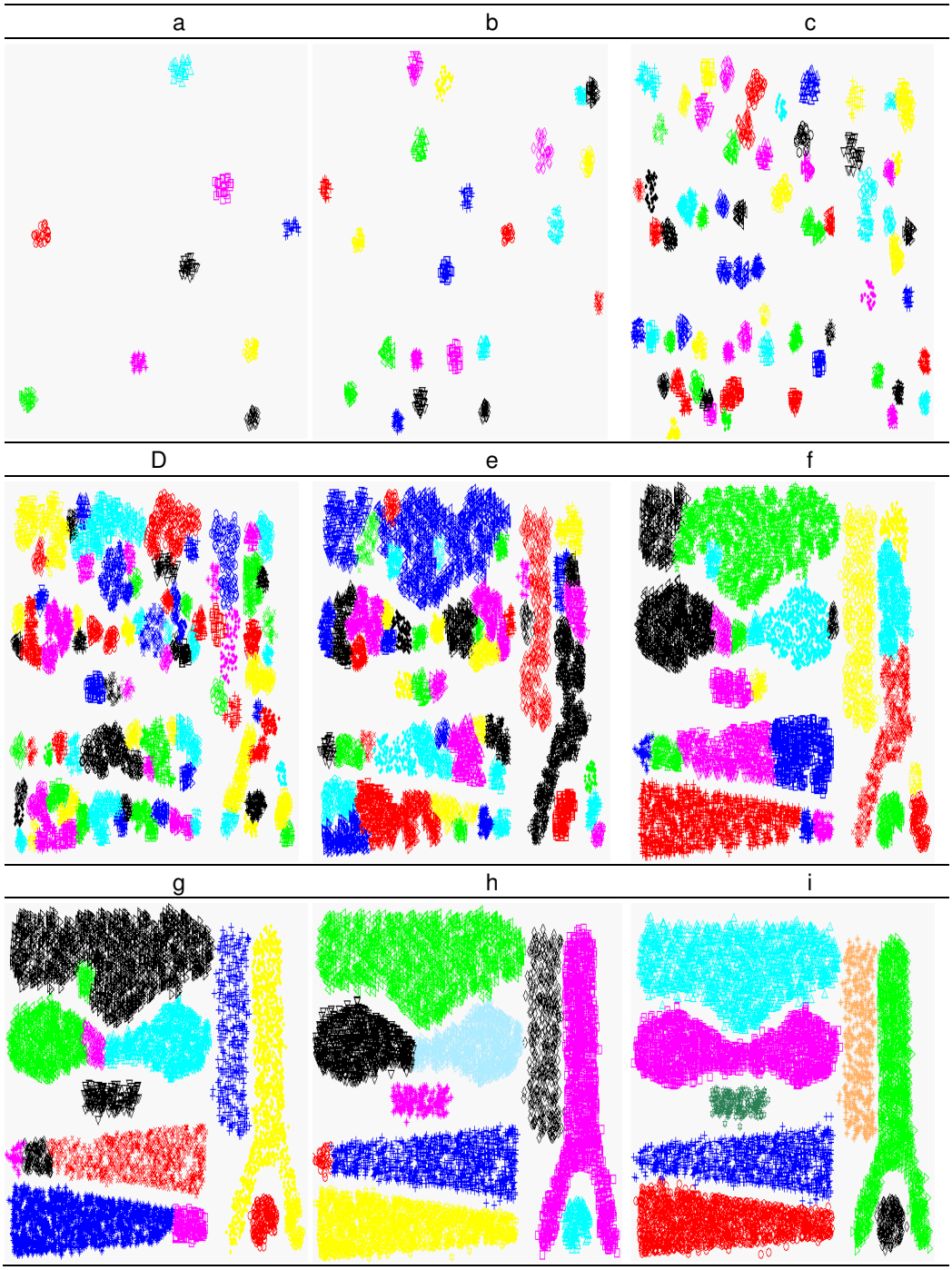


FIGURE 5: Nine snapshots of Adjusted CSHARP cluster propagation for the data set Chameleon DS5 at different iterations (a) 10, (b) 25, (c) 100, (d) 500, (e) 1000, (f) 2000, (g) 3000, (h) 4000 and (i) 5122.

3. MODIFIED CSHARP ALGORITHM

Figure 6 describes the modified CSHARP algorithm. Next, a logical data model and a graph-based interpretation of the algorithm are given. Finally, the time complexity of this algorithm is analyzed in section 3.2.

Input: Data points $P = \{p_1, p_2 \ldots, p_n\}$;
K {size of the neighborhood of a point};
T {threshold parameter for the size of a reference list} and
M {merge parameter for the size of the intersection between two reference blocks}.

Output: C set of generated clusters.

1: Construct similarity matrix S.

{Construct the Refer-To-List, K-NBp_i for each point $p_i \in P$}

2: K-NB$p_i = \{p_j \mid d(p_i, p_j) \leq d(p_i, p_k)\}$

{construct the Referred-By-List, RBp_i for each point $p_i \in P$.

3: **for all** $p_i \in P$ **do**
4: $V \leftarrow$ K-NBp_i
5: **for all** $v_j \in V$ **do**
6: **If** $p_i \in$ K-NBv_j **then**
7: $v_j \in$ RBp_i
8: **end if**
9: **end for**
10: **end for**

{From K-NBp_i and RBp_i, Construct the reference Lists RLp_i

11: **for all** $p_i \in P$ **do**
12: RLp_i= K-NB$p_i \cap$ RBp_i.
13: **end for**
14: Form a sorted (in a descending order) list L. L = $\{p_i | p_i$ is a representative point$\}$, based on the densities (i.e. |RLp_i|). Exclude the weak points from list L (those points for which |RLp_i| $< T$).
15: Sort the new list L' according to the homogeneity factors α_i's of the Reference-Lists RLp_i.

{Building Clusters}

16: $i \leftarrow 0$ {Initialize i}
17: $C_0 = \cup \{p_0,$ RL$p_0\}$
18: label point p_0 as belonging to cluster C_0.
19: label each point in RLp_0 as belonging to cluster C_0.
20: **While** $i \leq |L'|$ **do**
21: $i \leftarrow i + 1$ {increment i}
22: $C_i = \cup \{p_i,$ RL$p_i\}$
23: label point p_i as belonging to cluster C_i.
24: label each point in RLp_i as belonging to cluster C_i.
25: **if** $|\{p_i \cup$ RL$p_i\} \cap C_u| \geq M$, {where $u = 1,2, \cdots, i\text{-}1$} **then**
26: $C_i = \cup \{p_i,$ RL$p_i, C_u\}$
27: Update C_u labels, by marking each point in C_u as belonging to cluster C_i
28: **end if**
29: **end while**

FIGURE 6: CSHARP Clustering Algorithm

Note that although a reference list associated with a weak point does not participate in Cluster's growing, a weak point may itself belong to a Reference list of another strong point.

This implies that this weak point is not considered as noise.

3.1 Logical Data Model and Graph-based Interpretation

Figure 7 depicts the conceptual data model underlying the proposed algorithm. Two entities are shown; namely, the data sample and Reference-List associated with a data sample entities; as well as a one one-to-many" relation between them and one "many-to-many" relation associated with each of them.

- The relation “Has as member in its K-NB list” is not symmetric.
- The relation “Has as member” between a Reference List and its members is symmetric.
- The relation “Overlaps” is symmetric but not transitive.
- The entity "Reference List Associated with a Data sample"; say "r", is obtained as the intersection of two sets: the set of points having p in their K-Nearest Neighborhood and the K-Nearest Neighborhood of "p" itself.

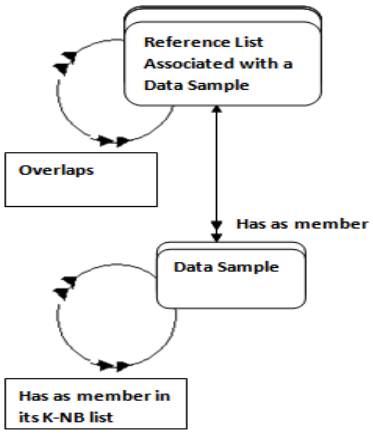


FIGURE 7: Conceptual data model.

Consider the following weighted graph (V, E), where a vertex in this graph corresponds to a Reference-List and an edge corresponds to an Overlaps relation between two Reference Lists. The weight of an edge corresponds to the cardinality of the intersection set between two Reference lists. Now, the effect of the CSHARP algorithm can be viewed as follows: Edges with weights less than the threshold M are removed. This decomposes the graph into independent components. The remaining connected vertices (i.e. vertices connected by edges having weights greater than M) are combined, the union of the data samples belonging to their reference lists correspond to the obtained clusters.

3.2 Time Complexity

The time complexity for computing the similarity matrix is $O(N^2)$, where N is the number of data points. This can be reduced to O(N log N), by using a data structure such as a k-d tree [1] or an R-tree[6]. The space complexity for t computing distances between samples is O(NF) where N is the number of data points and F is the number of features (or dimensions).

The time complexity of the algorithm can be analyzed as follows:

- line 2, finding k-nearest neighbors(refer-to-list): k iterations through all N data points are needed, hence it has a complexity of O(KN)
- lines 3-10, finding referred-by-list: it has, also, a complexity of O(KN)
- lines 11-13, finding reference-points: for each data point, its k-nearest neighbors are searched for mutual neighborhood. As a binary search is adopted; its complexity is of O(K log K), thus, the overall complexity of this step is O(NK log k).
- Line 14, computing homogeneity factor procedure: has a complexity of $O(NK)$ as , k iterations are needed for each of the N data points .

- line 15, sorting data sets procedure: sorting has a complexity of O(N log N), as binary sort is used.
- lines 25-28, clusters overlapping procedure: detecting overlapping among clusters has a linear complexity of O(K),since we iterate through reference-blocks of processed points and detect previously labeled points.
- lines 20-29, clusters propagation procedure: this is the main procedure. It has a complexity of O(NK) , since we iterate through all N data points, running clusters overlapping procedure for each data point. Accordingly, the overall complexity is O(NK log N) .

The overall time complexity for the modified CSHARP algorithm is O(N K log N) where N is the number of data points and K is the number of nearest neighbors. Modified CSHARP has a space complexity of O(KN), where N is the number of data points and K is the number of nearest neighbors used

3.3 Anatomy of CSHARP vs. Jarvis-Patrick and SNN Algorithms

- In SNN, similarity between two points p and q is computed as the number of nearest neighbors they share. In contrast, CSHARP's similarity is computed as the number of reference points two blocks share.
- In contrast to Jarvis-Patrick and SNN, CSHARP is a block-to-block rather than point-to-point clustering.
- Jarvis-Patrick and SNN work with k-nearest neighborhood which corresponds to a non-symmetric relationship between data points. On the other hand, CSHARP relies on a symmetric relation between any point and its reference list points.
- Jarvis-Patrick and SNN use static k-nearest neighbor lists, while CSHARP starts with static k-nearest neighbor lists then turns them into dynamic Reference Lists.
- In Jarvis-Patrick and SNN, the association between points is one-to-one, while it is one-to many in CSHARP.
- Both SNN and CSHARP propagate clusters simultaneously and agglomeratively, while Jarvis-Patrick builds a similarity graph then decomposes it by removing the weakest links.

4. EXPERIMENTAL RESULTS

4.1 Datasets Used

4.1.1. 2-d Chameleon's DS5 data set

DS5 consists of 8,000 points. It is a mixture of clusters with different densities used by Chameleon algorithm to illustrate its efficiency in obtaining arbitrary density clusters. The aim is to classify the data into 8 clusters of different densities.

4.1.2. Eight Low Dimensional Datasets

- This is a well known database in the pattern recognition literature. The data set contains 3 classes of 50 instances each, where each class refers to a type of iris plant. One class is linearly separable from the others 2; the latter are NOT linearly separable from each other [16].
- The Synthetic Control Charts (SCC) data set; obtained from UCR repository [14]; it includes 600 patterns, each of 60 dimensions (time points).
- The pen digit character recognition data from the UCI repository [16], consists of 10992 patterns, each of 16 dimensions. Features (dimensions) are used to describe the bitmaps of the digit characters. The aim is to properly classify the digit characters to 10 classes from 0 to 9.
- Libras movement data set: The dataset contains 15 classes of 24 instances each, where each class references to a hand movement type. It consists of 360 patterns, each of 91 dimensions.
- The Breast Cancer Wisconsin Diagnostic data from the UCI repository, consists of 569 patterns, each of 30 dimensions. The aim is to classify the data into two diagnosis (malignant or benign).
- SPECT heart data set: The dataset describes diagnosing of cardiac Single Proton Emission Computed Tomography (SPECT) images. It consists of 267 patterns, each of 22 dimensions. Each of the patients is classified into two categories: normal and

abnormal

- The Protein localization data; obtained from the UCI repository; is the Ecoli data set with 336 proteins of seven dimensions. The aim is to properly classify it to eight classes.
- The Protein Localization Sites, obtained from the UCI repository, consists of 1484 patterns, each of 8 dimensions. This database contains information about a set of Yeast cells. The task is to determine the localization site of each cell by partitioning it into 10 classes of varying distribution.

4.1.3. Two High Dimensional Datasets

- Corel image features data set: This dataset contains image features extracted from a Corel image collection. 2074 images of 144 dimensions were selected in this experiment according to criteria discussed in [2].
- Arcene data set which consists of 200 patterns each of 10,000 features. The task is to distinguish cancer versus normal patterns from mass-spectrometric data. This is a two-class classification problem with continuous input variables. Arcene has been part of the NIPS 2003 feature selection challenge.

4.2 Cluster Validation

As described in [21], for a given data set, a clustering algorithm can always produce a partitioning whether or not a particular structure in the data really exists. Different clustering approaches usually yield different results. Even for the same algorithm, the selection of a parameter or the presentation order of the input patterns may affect the final results. Therefore, effective evaluation criteria are critically important to provide users with a degree of confidence in the obtained clustering results. These assessments should be objective and have no preferences to any algorithm. V-measure [24], Purity and Entropy [23] are used for the purpose for clustering validation. Noise is taken into consideration in the validation process. This reduces the indexes values than if a noise-free validation process is adopted. However, this presents a more accurate assessment.

4.3 Results and Performance Evaluation

To compare the results obtained by the Modified CSHARP with those obtained by other algorithms, the nine data sets presented above have been used.

4.3.1. Chameleon's DS5 data set.

Five algorithms are compared: K-means, DBScan, SNN, Chameleon, and mitosis, in addition to Modified CSHARP.

Due to the presence of different densities in the DS5 data set, DBScan either identifies the lower density cluster but merges the two neighboring higher density ones, or do not identify the lower density cluster, but identifies the higher density ones. DBScan at the parameters setting (ε = 10 and MinPts = 3) can only identify six rather than eight clusters. Similarly, SNN merges two clusters together due to its static nature derived from DBScan. Both DBScan and SNN , (Figure 8b, and d) were unable to obtain a good clustering solution. Modified CSHARP obtained the genuine clusters at the parameters setting given in Table 1b for K in the range [23...25], Figure 8f.

Mitosis has been able to obtain the genuine clusters at parameter settings (f = 2.15 and k = 2.5) (Figure 8e). Mitosis results were obtained after discarding outliers. Clusters of sizes less than 1% of the data size are identified as outliers. While Mitosis uses a static model for discarding noise, CSHARP [25] focuses on the detection of chains of dense connected regions (defined by strong points) possibly including non-dense regions ; represented by weak points; as explained in section 2.1. Hence, in CSHARP any point not taken into consideration during cluster's propagation is considered as noise. Table 2 gives the numbers and percentages of strong, weak and noise points found in all the investigated data sets.

TABLE 1: Adjusted CSHARP's range of parameters setting for DS5 data set.

K	T	M
23	[3...5]	7

23	19	[6,7]
23	22	[6…9]
24	[15…17]	7
24	18	6
24	22	[6...9]
24	23	[4...9]
25	[16…18]	[7,8]
25	22	[6...9]
25	23	[5...9]

4.3.2. Eight Dimensional Data

Five data sets have been used to compare the results obtained by Modified CSHARP to the ground truth as well as to the results obtained by DBScan, K-means, Mitosis and Spectral clustering (as a state-of- the art technique [2]) algorithms. Chameleon and SNN have not been included in these experiments, due to the difficulty of adjusting their parameters. The Euclidean distance has been adopted as a metric for all data sets. Numerous experiments (100 experiments at least per algorithm per dataset) have been done on each algorithm to obtain its best indexes' values. V-Measure, Purity, and Entropy have been used to evaluate all the above clustering algorithms.

TABLE 2: Number of strong, weak and noise points found in the tested data sets and their percentages ;at given parameters settings; relative to the size of the corresponding data set.

Dataset	size	Strong Points	Weak Points	Noise Points
Chameleon DS5 (K=24, T=18 and M=6)	8000	6399 (79.98%)	1601 (20.02%)	225 (2.81%)
Synthetic Control Charts (K=14, T=7 and M=4)	600	479 (79.83%)	121 (20.17%)	1 (0.17%)
Pen Digit Data (K=31, T=14 and M=9)	10992	8732 (79.44%)	2260 (20.56%)	253 (2.30%)
Breast Cancer Diagnostic (K=41, T=22 and M=14)	569	492 (86.47%)	77 (13.53%)	15 (2.64%)
Ecoli (K= 22, T=11 and M=8)	336	240 (71.43%)	96 (28.57%)	25 (7.44%)
Yeast (K= 44, T=25 and M=15)	1484	848 (57.14%)	636 (42.86%)	134 (9.03%)
Arcene (K= 11, T=3 and M=1)	200	180 (90.0%)	20 (10.0%)	4 (2.0%)
SPECT Heart (K= 30, T=7 and M=4)	267	222 (83.15%)	45 (16.85%)	18 (6.74%)
Libras Movement (K= 10, T=7 and M=3)	360	173 (48.06%)	187 (51.94%)	46 (12.78%)
Iris (K= 24, T=8 and M=9)	360	138 (92.00%)	12 (8.00%)	0 (0.00%)
Corel (K= 43, T=21 and M=13)	2074	1230 (59.31%)	844 (40.69%)	120 (5.78%)

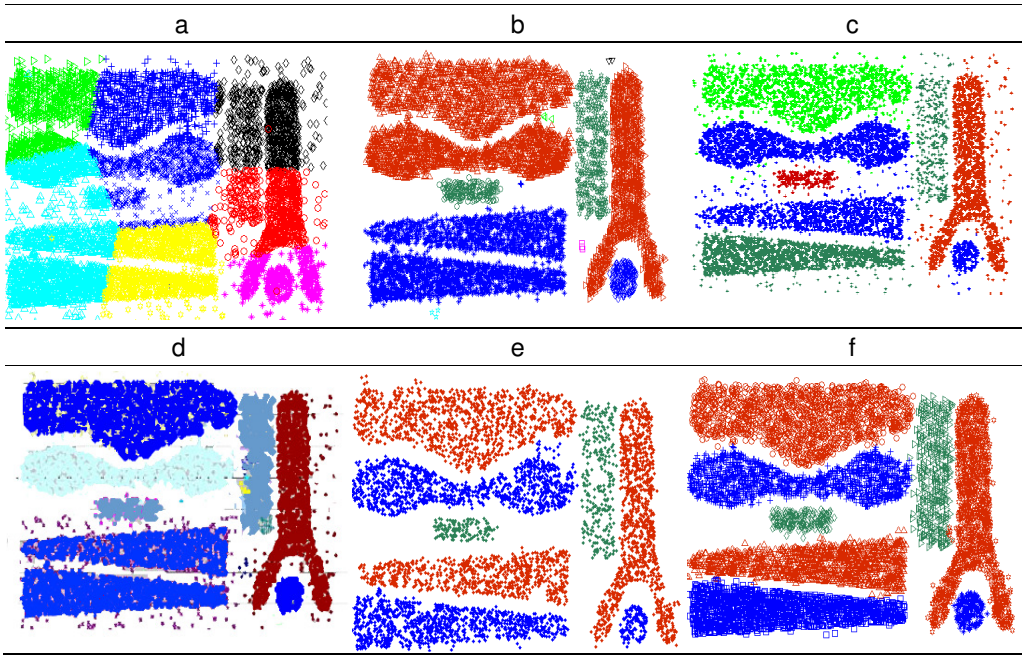


FIGURE 8: Results for the DS5 data set. Clusters are identified by a specific color, and a specific marker for: (a) K-means clusters; (b) DBScan clusters; (c) Chameleon clusters; (d) SNN clusters; (e) Mitosis clusters; and (f) Modified CSHARP clusters.

- Iris data set: Modified CSHARP and spectral clustering performed better for this data set, reached the highest indexes for F-measure and Purity and lowest index for entropy as shown in Figure 10a. The concept of relatedness fails with iris data set, since it consists of three classes, two of them are not linearly separable; thus, Due to the merging of these two clusters, Mitosis obtained the lowest indexes for this data set. Mitosis failed to detect the original three clusters and detected only two clusters. On the contrary, Modified CSHARP and Spectral clustering detected the three original clusters, Figure 11.
- Time series data: For SCC, The original six clusters are shown in Figure 9. All algorithms (with the exception of CSHARP) failed to discover the cluster labeled normal, due to its relatively low density. DBScan using the parameters setting (ε = 50 and MinPts = 3) detected the cyclic cluster. However, it merged the Up-Shift, and the Up-Trend clusters together, as well as the Down-Shift and the Down-trend clusters. K-means, at $k = 6$ merged patterns from different clusters together. The values of the V-Measure, Purity and Entropy are shown in Figure 10b. Due to the discovery of the Normal low dense cluster; at the parameters setting (K=11, T=5 and M=3); CSHARP reached the maximum values for both V-measure and Purity indexes and the lowest for Entropy index, Figure 12.

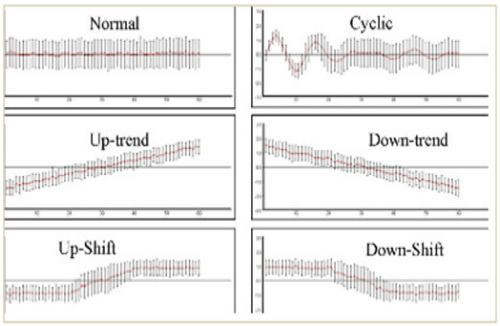
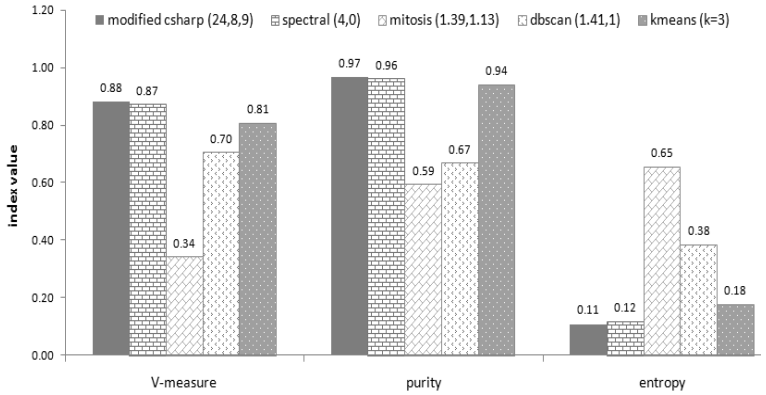


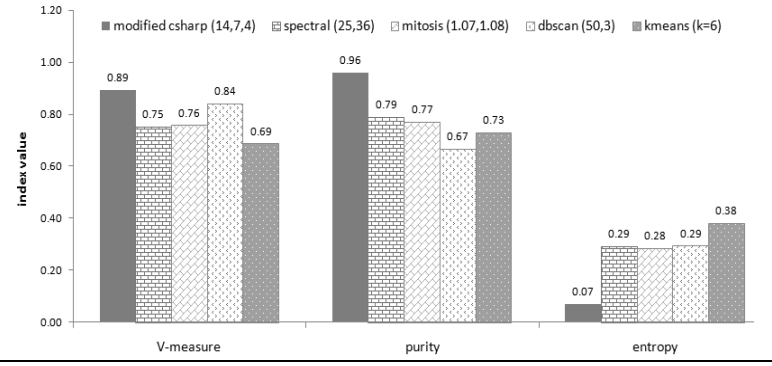
FIGURE 9: Time series clusters (mean ± standard deviation) of SCC data original classes.

- Pen Digits data set: CSHARP gave good solution and gave a very high Purity and low Entropy relative to the other solutions as shown in Figure 10c. DBScan and K-means combined several characters in one cluster. For instance, K-means combined 5 with (8, 9) and 1 with (2, 7), DBScan combined (1,2) and (5,9) and Mitosis combined 1 with (2, 7) and (5,9).
- Disease diagnosis data set: For breast cancer diagnosis, CSHARP obtained the maximum values for all the indexes at the parameters setting (K=50, T=8 and M=5). Figure 13a gives the corresponding validity indexes values.
- SPECT heart and Libras Movement : Adjusted CSHARP performed better for all indexes as shown in Figures 13b and c respectively.
- Ecoli and Yeast: The values of the indexes are given in Figures 13e and f, respectively showing the superiority of CSHARP over all other algorithms.

a



b



c

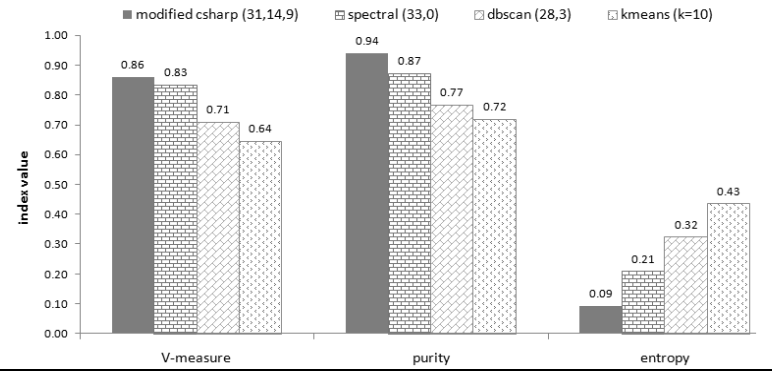


FIGURE 10: V-Measure, Purity and Entropy for: (a) Iris data set, and (b) SCC time series, and (c) pen digit recognition data set.

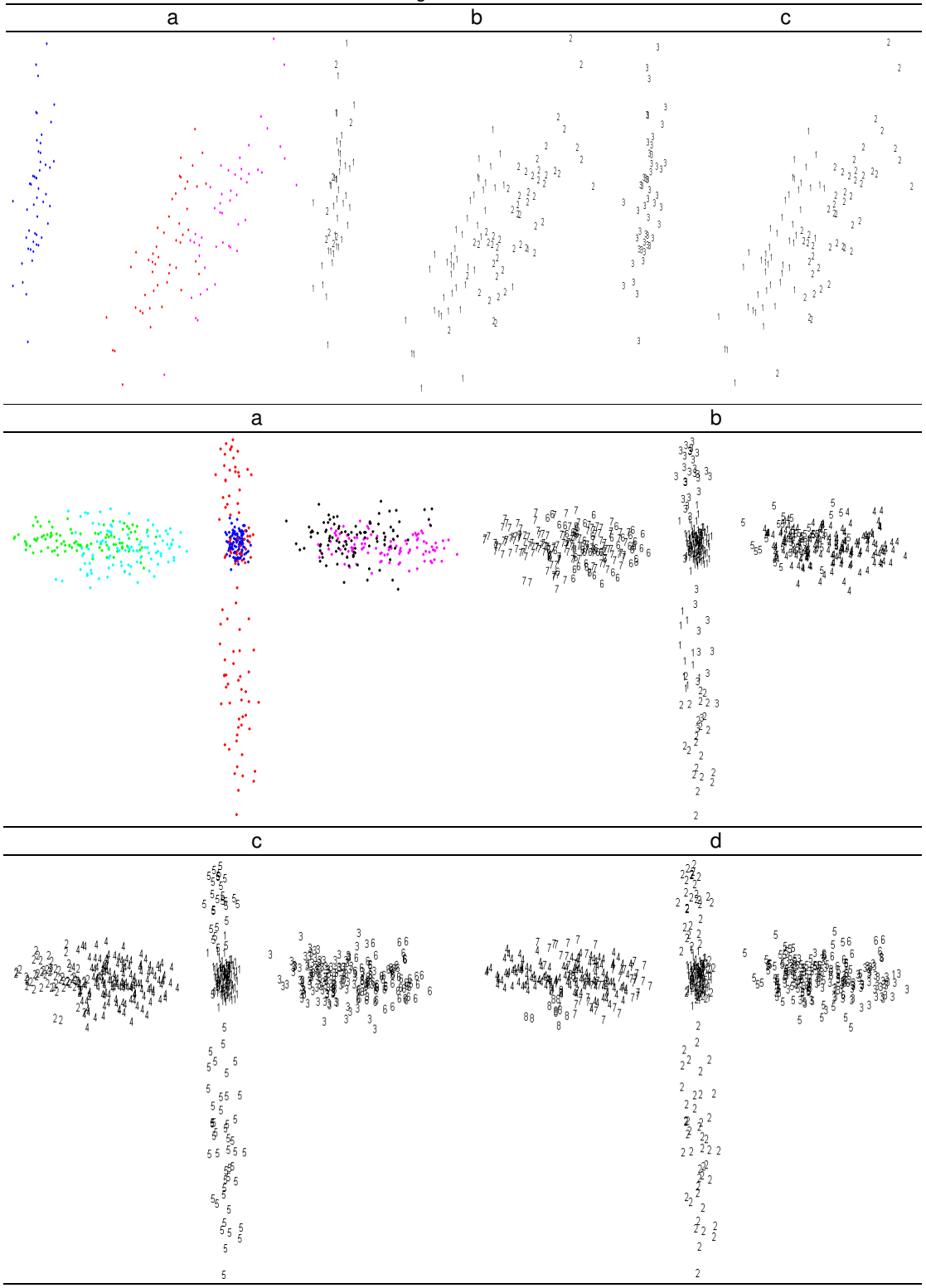
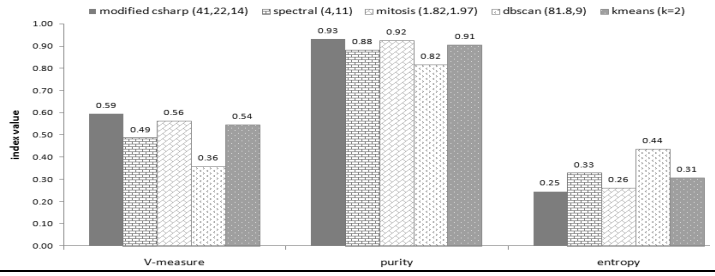


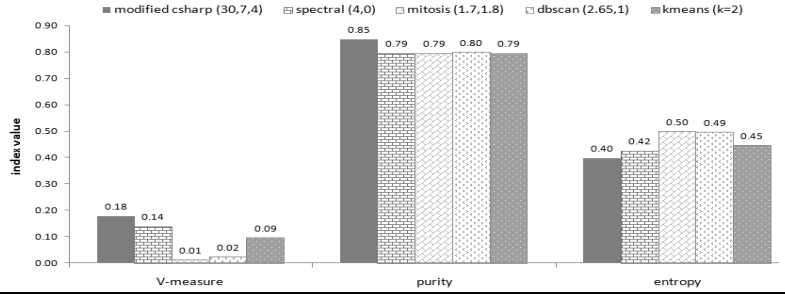
FIGURE 12: Plotting SCC time series data set using PCA for (a) reference solution, (b) Mitosis, (c) Spectral clustering, and (d) Modified CSHARP.

FIGURE 11: Plotting Iris data set using PCA for (a) reference solution, (b) Mitosis, and (c) Modified CSHARP and Spectral clustering.

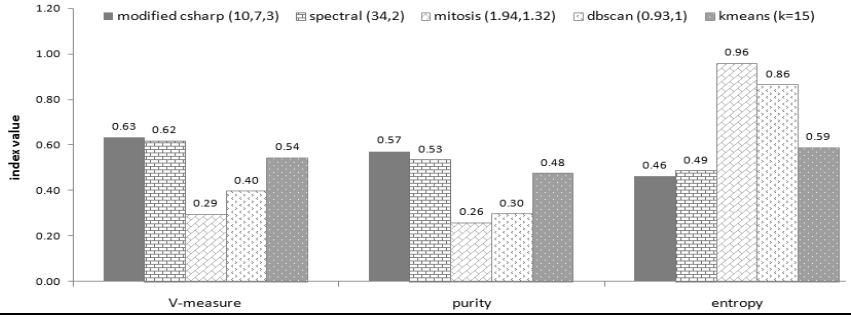
a



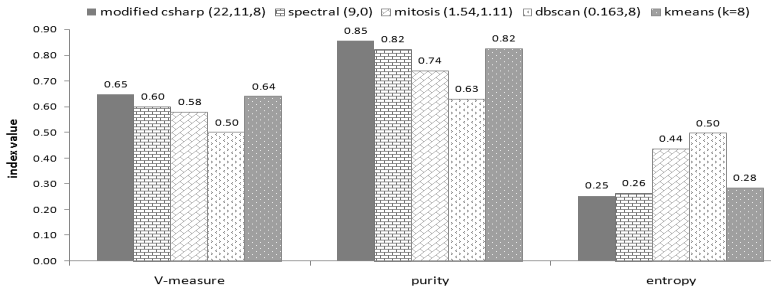
b



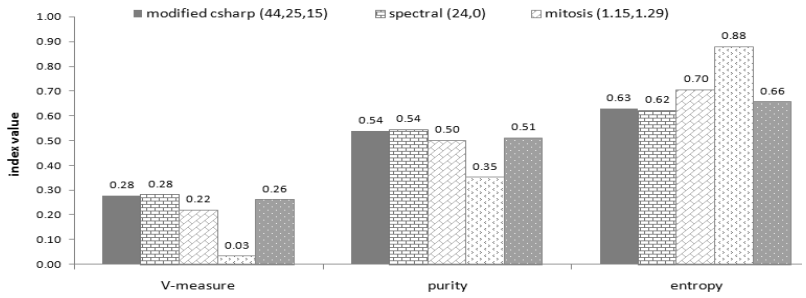
c



d



e



a

.**FIGURE 13**: V-Measure, Purity and Entropy for: (a) Breast cancer Wisconsin diagnostic data set, (b) SPECT heart data set, and (c) Libras movement data set.(d) Ecoli data set, and (e) Yeast data set.

4.3.3. High Dimensional Data

To investigate the performance of the Modified CSHARP when applied to high dimensional data sets, two data sets were used from the 2003 Nips Feature extraction challenge [7]. Both CSHARP and spectral clustering obtained competitive results as shown in Figures 14a and b for Corel and Arcene data sets, respectively, with CSHARP performing slightly better for all indexes. Spectral clustering algorithm implemented by [2] uses two parameters, "K" for the number of K-nearest neighbors; used to generate a sparse symmetric distance matrix and σ value used in similarity function S,

$$\text{where } S_{ij} = \exp\left(-\frac{\|x_i - x_j\|^2}{2*\sigma_i \sigma_j}\right).$$

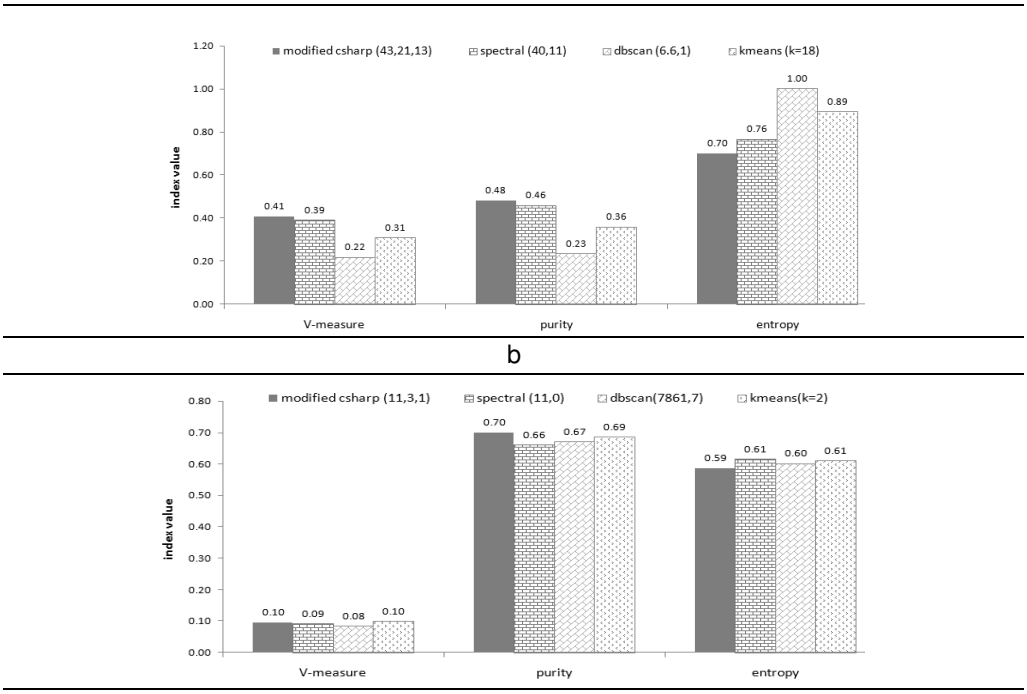


FIGURE 14: V-Measure, Purity and Entropy for: (a) Corel data set and (b) Arcene data set.

In this section, the efficiency of Modified CHARP is recorded when compared with well-known clustering algorithms such as K-means, DBScan, Chameleon, Mitosis, and Spectral Clustering. V-measure, Purity, and Entropy are used showing the superiority of CSHARP over the other tested algorithms for almost all used data sets .

Modified CSHARP succeeded to overcome the limitations that the other algorithms suffer from. It can deal with classes of different densities whereas DBScan cannot, deal with arbitrary shapes whereas K-means cannot, deal with arbitrary cluster's sizes where spectral clustering cannot, and deal with interlaced (overlapped) clusters where Mitosis cannot. Moreover, it can scale easily whereas Chameleon cannot. Therefore, it can be said that the proposed technique is less likely to merge clusters of different densities or different homogeneities as indicated by the obtained results. Next, the performance of Modified CSHARP relative to the other algorithms is investigated.

4.4 Speed Performance

The speed of Modified CSHARP has been compared to the speed of CSHARP, Chameleon and DBScan as shown in Figure. 15, using the DS5 data set, after dividing it into data subsets, each of size 1000 patterns. The subsets are added incrementally, and the speed of the algorithm is recorded for each increment. The time considered is the time required for running the core clustering algorithms, excluding the time for obtaining the similarity matrix between the sample points. The time is measured in seconds. The average running times are computed for DBScan, K-means, and Modified CSHARP for 10 runs, with standard deviation listed in Table 3. Chameleon was tested only once for each increment. K-means was iterated 100 times for each experiment to provide better convergence.

The adopted algorithms as well as the proposed Adjusted CSHARP algorithm have been executed on a machine with the following configuration: 3.00 GHz processor, 1.0 GB RAM, and running Linux operating system (Ubuntu 10.04 LTS).

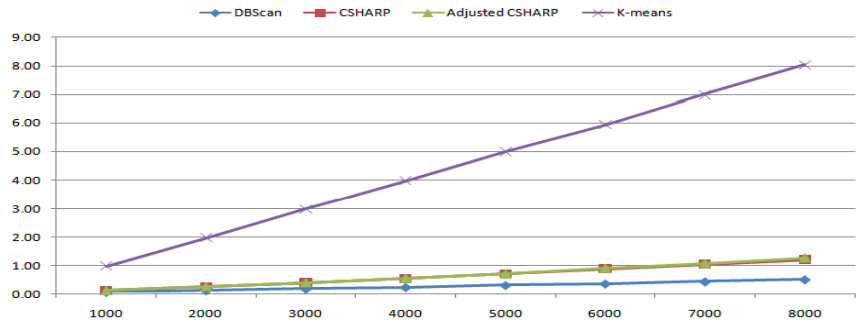
TABLE 3: Standard Deviation for 10 Runs on Adjusted CSHARP, DBScan, and K-means on Chameleon's DS5.

Data Size	1000	2000	3000	4000	5000	6000	7000	8000
K-means	0.020	0.008	0.017	0.015	0.021	0.018	0.023	0.043
DBScan	0.008	0.012	0.017	0.025	0.028	0.021	0.017	0.034
Adjusted CSHARP	0.013	0.013	0.014	0.016	0.016	0.019	0.025	0.037

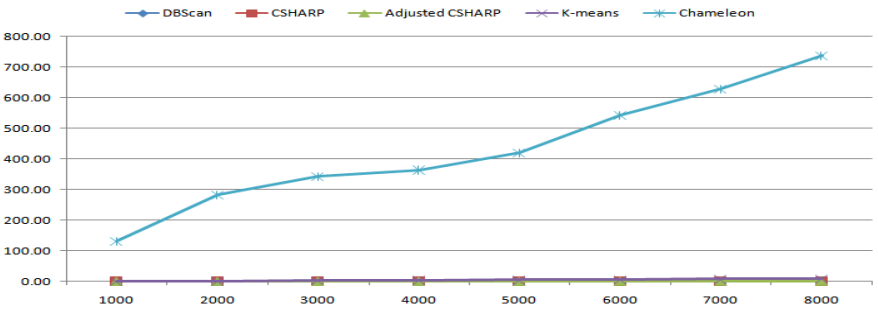
5. CONCLUSION

In this paper, a modified version of the novel shared nearest neighbors clustering algorithm, CSHARP[25] has been presented. Density and homogeneity are combined in a new homogeneity factor used to order the merging of blocks of points . It can find clusters of varying shapes, sizes and densities; even in the presence of noise and outliers and in high dimensional spaces as well. It is based on the idea of letting each data point vote for its K-nearest neighbors and adopting the points with the highest votes as clusters' seeds. Two clusters can be merged if their link strength is sufficient. Any data point not belonging to a

a



b



c

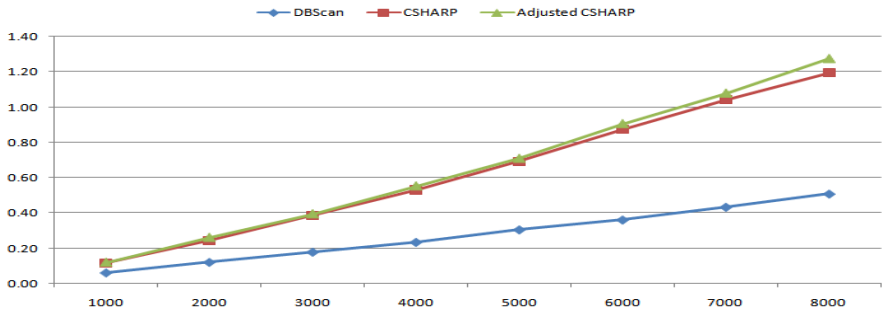


FIGURE. 15: Speed of Adjusted CSHARP and Adjusted CSHARP using Chameleon's DS5 data set, compared to (a) DBScan and K-means (b) DBScan, Kmeans, and Chameleon (c) DBScan.

cluster is considered as noise. The results of our experimental study on several data sets are encouraging. A wide range of possible parameters settings yield satisfactory solutions using the validation indexes adopted in [8], [9] and [10]. Adjusted CSHARP solutions have been found, in general, superior to those obtained by DBScan, K-means and Mitosis and competitive with spectral clustering algorithm adopted in[2].

More work is needed to introduce a procedure for parameters selection. Also, we intend to investigate the behavior of our clustering approach on other types of data. Moreover, we intend to parallelize our algorithm as its clustering propagation is inherently parallel, as has been shown in section 2.2. Finally, we have made the algorithm publicly available to other researchers at http://www.csharpclustering.com.

ACKNOWLEDGMENT

Authors would like to thank Dr. Noha A. Yousri for several useful discussions. Noha A. Yousri is with the Department of Computer and System Engineering, Faculty of Engineering, Alexandria University, Alexandria, Egypt.

REFERENCES

[1] Bentley, J. L. (1975). Multidimensional binary search trees used for associative searching. Communications of the ACM, 18(9):509–517.

[2] Chen, W.-Y., Song, Y., Bai, H., Lin, C.-J., and Chang, E. Y. (2011). Parallel spectral clustering in distributed systems. IEEE Trans. Pattern Anal. Mach. Intell, 33(3):568–586.

[3] Ding, C. H. Q. and He, X. (2004). K-nearest-neighbor consistency in data clustering: incorporating local information into global optimization. In Haddad, H., Omicini, A., Wainwright, R. L., and Liebrock, L. M., editors, SAC, pages 584–589. ACM.

[4] Ertoz, Steinbach, and Kumar (2003). Finding clusters of different sizes, shapes, and densities in noisy, high dimensional data. In SIAM International Conference on Data Mining, volume 3.

[5] Ester, M., Kriegel, H.-P., Sander, J., and Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In KDD, pages 226–231.

[6] Guttman, A. (1984). R-trees: A dynamic index structure for spatial searching. In ACM SIGMOD. Also published in/as: UCB, Elec.Res.Lab, Res.R. No.M83-64, 1983, with Stonebraker,M.

[7] Guyon, I., Gunn, S., Nikravesh, M., and Zadeh, L. (2006). Feature Extraction: Foundations and Applications. Springer Verlag.

[8] Halkidi, M., Batistakis, Y., and Vazirgiannis, M. (2002). Cluster validity methods: part I. SIGMOD Record (ACM Special Interest Group on Management of Data), 31(2):40–45.

[9] Hammouda, K. M. and Kamel, M. S. (2002). Phrase-based document similarity based on an index graph model. In ICDM, pages 203–210. IEEE Computer Society.

[10] Hubert, L. and Arabie, P. (1985). Comparing partitions. Journal of Classification, 2:193–218.

[11] Jain, A. K. and Dubes, R. C. (1988). Algorithms for Clustering Data. Prentice Hall, Englewood Cliffs.

[12] Jarvis, R. and Patrick, E. (1973). Clustering using a similarity measure based on shared near neighbors. IEEE Transactions on Computers, 22(11):1025–1034.

[13] Karypis, G., Han, E.-H. S., and NEWS, V. K. (1999). Chameleon: Hierarchical clustering

using dynamic modeling. Computer, 32(8):68–75.

[14] Keogh, E. J., Xi, X., Wei, L., and Ratanamahatana, C. A. The ucr time series classification/clustering, homepage: (www.cs.ucr.edu/ eamonn/time_series_data/), 2006.

[15] Lee, J.-S. and Ólafsson, S. (2011). Data clustering by minimizing disconnectivity. Inf. Sci, 181(4):732–746.

[16] Murphy, P. M. and Aha, D. W. (1992). UCI repository of machine learning databases. Machine-readable data repository, University of California, Department of Information and Computer Science, Irvine, CA.

[17] Nadler, B. and Galun, M. (2006). Fundamental limitations of spectral clustering. In Schölkopf, B., Platt, J. C., and Hoffman, T., editors, NIPS, pages 1017–1024. MIT Press.

[18] Rand, W. M. (1971). Objective criteria for the evaluation of clustering methods. American Statistical Association Journal, 66(336):846–850.

[19] Rijsbergen, C. J. V. (1979). Information Retrieval, 2nd edition. Butterworths, London.

[20] von Luxburg, U. (2007). A tutorial on spectral clustering. Statistics and Computing, 17(4):395–416.

[21] Xu, R. and II, D. C. W. (2005). Survey of clustering algorithms. IEEE Transactions on Neural Networks, 16(3):645–678.

[22] Yousri, N. A., Kamel, M. S., and Ismail, M. A. (2009). A distance-relatedness dynamic model for clustering high dimensional data of arbitrary shapes and densities. Pattern Recognition, 42(7):1193–1209.

[23] Ying Zhao and George Karypis. 2001. Criterion functions for document clustering: Experiments and analysis. Technical Report TR 01.40, Department of Computer Science, University of Minnesota

[24] Andrew Rosenberg and Julia Hirschberg, 2007. V--Measure: a conditional entropy--based external cluster evaluation measure. EMNLP '07

[25] Mohamed A. Abbas Amin A. Shoukry, and Rasha F. Kashef, ICDM 2012. CSHARP: A Clustering Using Shared Reference Points Algorithm, International Conference on Data Mining, Penang, Malaysia, World Academy of Science, Engineering and Technology.

Query Processing with K-Anonymity

Mohamed Y. Eltabakh
Department of Computer Science meltabakh@cs.wpi.edu
Worcester Polytechnic Institute
Worcester, MA, USA, 01604

Jalaja Padma
Cisco Systems jpadma@cisco.com
San Jose, California, USA, 95134

Yasin N. Silva
Division of Mathematical & Natural Sciences ysilva@asu.edu
Arizona State University
Tempe, Arizona, USA, 85281

Pei He, Walid G. Aref, Elisa Bertino
Department of Computer Science {phe, aref, bertino}@cs.purdue.edu
Purdue University
West Lafayette, Indiana, USA, 47907

Abstract

Anonymization techniques are used to ensure the privacy preservation of the data owners, especially for personal and sensitive data. While in most cases, data reside inside the database management system; most of the proposed anonymization techniques operate on and anonymize isolated datasets stored outside the DBMS. Hence, most of the desired functionalities of the DBMS are lost, e.g., consistency, recoverability, and efficient querying. In this paper, we address the challenges involved in enforcing the data privacy inside the DBMS. We implement the k-anonymity algorithm as a relational operator that interacts with other query operators to apply the privacy requirements while querying the data. We study anonymizing a single table, multiple tables, and complex queries that involve multiple predicates. We propose several algorithms to implement the anonymization operator that allow efficient non-blocking and pipelined execution of the query plan. We introduce the concept of k-anonymity view as an abstraction to treat k-anonymity (possibly, with multiple k preferences) as a relational view over the base table(s). For non-static datasets, we introduce the materialized k-anonymity views to ensure preserving the privacy under incremental updates. A prototype system is realized based on PostgreSQL with extended SQL and new relational operators to support anonymity views. The prototype system demonstrates how anonymity views integrate with other privacy-preserving components, e.g., limited retention, limited disclosure, and privacy policy management. Our experiments, on both synthetic and real datasets, illustrate the performance gain from the anonymity views as well as the proposed query optimization techniques under various scenarios.

Keywords: Data Privacy, K-Anonymity, Query Processing, Database Management Systems

1. INTRODUCTION

Privacy preservation is a key requirement for organizations holding personal or sensitive data. Organizations need to comply with the privacy preferences of data owners and privacy laws that regulate the use and sharing of such data [2]. Examples of privacy laws include the United States Privacy Act of 1974, the Australian Privacy Amendment Act of 2000, and The Health Insurance Portability and Accountability Act of 1996. These requirements have motivated a significant amount of work to answer the challenging question: How to make use of the data, e.g., querying and analysing, while ensuring the required level of privacy of data owners? Data anonymization techniques, e.g., [11,

12, 16, 17], have proposed to achieve the privacy preservation by ensuring that the identity of each individual cannot be distinguished from a group of other individuals whose records exist in the same dataset. K-anonymity [16, 17] is one of the foremost anonymization techniques proposed in literature. It ensures that the identity of each individual is hidden within a group of at least k-1 individuals. Clustering and generalization, e.g., [5, 15], are common approaches to implement k-anonymity as we will discuss in more detail in Section II. Several algorithms have been proposed to provide stronger privacy preservation over k-anonymity, e.g., l-diversity [12], and t-closeness [11]. However, most of these techniques are standalone anonymization techniques implemented at the application level. That is, they assume the data is exported outside the database system and stored externally as standalone datasets. Moreover, the anonymized version of the data is also stored outside the database system and released to various Applications. This approach has major drawbacks including: (1) Sensitive data is transferred from the database system to external applications, which may compromise the privacy of the data owners, (2) Most desirable functionalities of the DBMS are lost, e.g., consistency, recoverability, efficient querying, and authorization mechanisms, (3) The entire dataset needs to be anonymized even if users are interested in a single (or few) records, and (4) Anonymizing complex queries with multiple predicates and joined tables is not feasible in the standalone version.

In this paper, we propose extending the database system to support the privacy-preservation requirements from within the database engine to overcome the shortcomings discussed above. We implement the k-anonymity algorithm as a relational operator that can compose, along with the standard query operators, a privacy-aware query plans. We propose several query optimizations, e.g., pushing the selection operator below the anonymization operator, to build efficient query plans. We propose several algorithms, block-level and tuple-level, to implement the anonymization operator and allow efficient non-blocking and pipelined execution of the query plan. We introduce the notion of anonymization views (A-views, for short) to abstract the problem of anonymization as a relational view on the base tables containing sensitive data. We extend the k-anonymity to multi-k-anonymity to support personalized anonymization, i.e., different individuals may choose different k values. We study anonymizing a single table, multiple joined tables, and complex queries that involve multiple predicates. We also address several challenges that emerge when anonymizing complex queries and/or joined tables (which they can be anonymization views themselves). In the paper, we focus on implementing the k-anonymity algorithm, however, the proposed anonymization view is independent of the underlying anonymization technique. We realized the proposed system through extensions to PostgreSQL where we extended SQL to create and manipulate the anonymization views and introduced a new anonymization operator to the query engine.

The rest of the paper is organized as follows. In Section II, we discuss related work. In Section III, we present the architecture of the proposed open-source privacy-aware database system. In Sections IV, we introduce the needed definiens and concepts. In Sections V, VI, and VII, we present the logical and materialized A-views over single and multiple tables. Sections VIII presents the performance evaluation and experimental results. Finally, Section IX contains concluding remarks.

2. RELATED WORK

Data anonymization techniques, e.g., [11, 12, 16, 17], have been proposed to enforce privacy preservation requirements. K-anonymity [16, 17] is one of the most common anonymization techniques in literature. It ensures that the identity of each individual cannot be distinguished from at least a group of other k-1 individuals in the same dataset. In this technique, the attribute-combinations that may reveal the identity of an individual are called Quasi-Identifiers (QI, for short) while the attributes pertaining to sensitive values, e.g., the disease name, are called Sensitive-Attributes (SA, for short). K-anonymity algorithms can be implemented using clustering or generalization techniques, e.g., [5, 15]. In the clustering approach, clustering algorithms are employed to identify groups of similar records that are represented by a single record. In the generalization technique, each quasi-identifier attribute is associated with a domain generalization hierarchy (DGH) from which the QI values can be generalized to form groups of at least k tuples with identical QI values. Examples of DGHs and k-anonymized tables are given in Figs. 1 and 2, respectively. Fig. 1 shows three attributes, Disease, YearOfBirth, and ZipCode, associated with their DGHs. Fig.2 shows a list of patient records where the combinations of

QI attributes (Birthdate and Area) are 2-level anonymous, i.e., there are at least 2 identical records (w.r.t QI values) for every QI attribute combinations.

Several algorithms have been proposed to provide stronger privacy-preservation over the k-anonymity technique, e.g., l-diversity [12], and t-closeness [11]. The t-closeness technique ensures that the distribution of sensitive values in a single anonymized group is as close as possible to the distribution in the base table. More sophisticated anonymization algorithms can also be embedded. These data anonymization techniques, as discussed in Section I, are standalone techniques that operate on the data at the application layer outside the database management systems. In contrast, the proposed materialization views ensure the data privacy inside the DBMS, and hence fully utilize the functionalities and query processing power of the DBMSs. Moreover, the abstraction of anonymization views is independent of the underlying anonymization algorithm being used. For example, and as a proof of concept, we implement the t-closeness algorithm in Datafly system [16] on top of the k-anonymity technique.

The concept of personalized privacy in [19] allows data owners to choose the level of generalization of sensitive attribute and to integrate it with k-anonymity to produce a stronger anonymized version of the data. We support sensitive attribute generalization. However, we keep generalization independent of anonymization, and provide it as an additional technique for protecting the privacy of sensitive data. While the integration of SA generalization with anonymization provides better utility, it may suffer from information leak through the locality of sensitive data. For example, as presented in [19], a tuple whose sensitive attribute is generalized to its parent value in the DGH containing 3 children is considered 3-anonymous, whereas in regular anonymization, the tuple can be part of a QI group having sensitive attributes that could be distributed across all the leaves in the DGH.

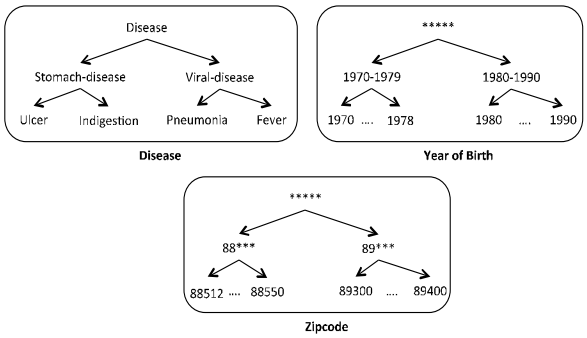


FIGURE 1: Domain Generalization Hierarchies.

Birthdate(QI)	Sex(QI)	Area(QI)	Disease(SA)
1970-74	F	Phoenix	Ulcers
1970-74	F	Phoenix	Indigestion
1985-89	M	Arkansas	Indigestion
1985-89	M	Arkansas	Fever
1985-89	M	Arkansas	Flu
1970-80	F	USA	Ulcers
1970-80	F	USA	Flu

FIGURE 2: An example of a 2-anonymous table.

Supporting data privacy at the database-system level is not a new approach. For example, Hippocratic Database [2] is an extension to the standard DBMS to manage the privacy of the data inside the database systems. Hippocratic databases include several privacy components to support policy management, limited disclosure, limited collection, limited retention, and compliance, among others. When a query is issued, the Hippocratic database system ensures that the recipient of the query answer receives only the data (s)he is allowed to see as per the policies. A data model to enforce the limiting disclosure principle enunciated in Hippocratic databases is proposed in [10]. In [3, 6], a Hippocratic fine-grained access control is proposed to limit the data disclosure. In [9], a middleware system is designed to implement and integrate several Hippocratic components. Other extensions to Hippocratic databases have been proposed in [1, 4, 10]. However, Hippocratic databases do not address data anonymization as a mechanism for privacy preservation, and they do not study the privacy as an operator inside the database engine that interacts with the other query operators. Most extensions to Hippocratic databases have been at the application-level or middleware-level between the application and the database. Anonymization views differ from the above proposals in that they are fully integrated inside the database.

Multi-relational k-anonymity [13] highlights several privacy attacks possible in multi-relational scenario and proves that the algorithms for a single relation do not achieve anonymity in the case of multiple relations. In this paper, we address the issue of anonymizing joined tables and propose different semantics for query results to ensure data privacy. For example, while [13] proposes hiding every tuple in some scenarios, we propose reporting false-positive tuples in addition to the correct (true-positive) tuples to ensure the privacy while maintaining high utility of the query result. Metrics to measure the privacy and utility of anonymized data are discussed in [20]. We adopt the Normalized Certainty Penalty proposed in [20] as a measure of utility.

3. SYSTEM ARCHITECTURE

The architecture of the proposed privacy-preserving database system is given in Figure 3. The system is an extension to our Hippocratic PostgreSQL system published in [21] which only supports a simple case of single-table anonymization. The goal is to develop a complete privacy-aware open-source DBMS by integrating the hippocratic database components, e.g., Policy Manager, Limited Disclosure, and Limited Retention, with the proposed anonymization techniques. The Policy Translator component is used to translate a P3P policy [7] into metadata. The Parser integrated with limited disclosure and retention support uses this translated metadata to produce query plans. We needed to change the Parser, Planner, and Executor of the database engine to integrate anonymization-related changes. We integrate and extend the ideas in [2, 9, 10] to implement our limited disclosure module. We use keywords PURPOSE and RECIPIENT to associate a purpose and a recipient to a given query, for example:

```
SELECT p.name, p.birth, p.sex, p.disease
FROM patient p PURPOSE research RECIPIENT lab;
```

Even though the definition of limited retention in Hippocratic databases [2] is to retain the information only as long as required, deletion of information after this period is not always trivial. Our approach to limited retention is through privacy policy specification of the retention periods for various attributes by each data owner. Every query complies with the policy for retention and does not display data that has exceeded its retention period. The limited retention and limited disclosure components are described in detail in [21]. The anonymization support and its interactions with the above privacy preserving modules are explained in the next sections.

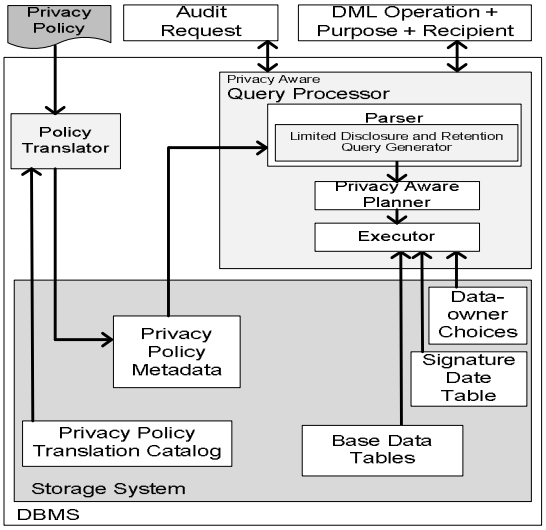


FIGURE 3: Hippocratic PostgreSQL architecture

4. DEFINITIONS AND CONCEPTS

Anonymization views (A-views) can be defined on a single table, multiple tables, or arbitrary SQL queries. They can be logical or materialized. A-views are similar to the standard database views except

that they employs anonymization operators during query processing to ensure that the anonymization requirements are met. Before we describe how A-views work, we present the needed definitions and concepts.

Multi-k-Anonymity: The concept of k-anonymity is introduced to protect the privacy of individuals [16, 17]. It associates one global privacy measure k to all individuals. However, in order to be applicable to a wide range of applications and to support personalized privacy, we need to extend k-anonymity to support multi-k-anonymity. That is, every individual may specify his/her preference for k. Furthermore, we allow the data owners to specify a k value for each purpose/recipient combination. These preferences are part of the privacy policy signed by the data owners and are kept inside the database. The flexible multi-k-anonymity model is highly motivated by the diverse privacy requirements of the data owners. For example, a hospital database may contain information about ordinary and well-known individuals who, most probably, have different privacy constraints.

Domain Generalization Hierarchy (DGH): The k-anonymity algorithm uses domain generalization hierarchies to generalize the values in certain attributes. Example of DGH is depicted in Fig. 4. In our system, these DGHs are stored in a database table. A single DGH can be associated with one or more columns in the database. To create and populate the DGHs, we introduce the following extended SQL commands:

CREATE DGH <dgh_name>;
INSERT INTO DGH <dgh_name> <child, parent>;

The association between the DGHs and the database columns is established when creating the anonymization views.

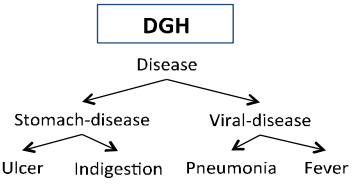


Table Data

QI	SA
A1	Ulcer
A2	Indigestion
B1	Ulcer
B2	Fever

Probability Distribution

SA	Probability
Ulcer	0.5
Indigestion	0.25
Fever	0.25
Pneumonia	0

FIGURE 4: DGH, Table, and Sensitive Attributes Distribution

```
CREATE [MATERIALIZED] ANONYMIZATION_VIEW a_view_name
ON query
WITH ANONYMIZATION_ID id_col
ANONYMIZATION_QUASI_ID ({qi_col [DGH_NAME dgh_name]} [,..])
ANONYMIZATION_SENSITIVE_ATTR ({sa_col [DGH_NAME dgh_name]} [,..])
profile_col REFERENCES k_profile_table (k_col [,sa_level_col])
```

FIGURE 5: Create Anonymization View command

Sensitive Attribute (SA) Generalization: Our work adopts the idea of sensitive-attribute generalization proposed in [9, 19] to provide additional privacy to data owners. However, unlike [19], this feature is

kept independent of the anonymization process. That is, apart from specifying the k values, the data owner may specify the level of generalization (in DGH) of the sensitive attribute(s) for every purpose/recipient combination. For example (refer to Figure 4), if a person suffering from 'Ulcer' specifies a generalization level of 1, then the Disease attribute (the SA attribute) is generalized to 'stomach-disease', otherwise, it is reported as is.

Anonymization View (A-view): We introduce a new SQL construct to define an anonymization view as presented in Fig. 5. The command declares the identifiers, quasi-identifiers, and sensitive attributes of the data to be anonymized. Additionally, it associates DGHs to the quasi-identifiers. The profile_col gives the provision of choosing any attribute as the foreign key to a table containing the k-values and sensitive attribute generalization levels for various combinations of purpose and recipients. Anonymization views can be either logical or materialized (if the keyword MATERIALIZED is used in the command). Logical A-views suffer from temporal attacks if the datasets are not static [17]. This is because updating the data will results in different anonymized dataset each time. Hence, in the case of non-static datasets, we materialize the A-views to prevent these temporal attacks under incremental updates. Details about materialized A-views are discussed in Section VII.

5. ANONYMIZATION VIEW ON SINGLE TABLE

We propose two different query plans to anonymize queries on single tables. The two plans differ in where to plug-in the anonymization operator w.r.t the selection operator. The anonymization operator implements an extended version of the Datafly algorithm in [16] to accommodate for multi-k-anonymity and t-closeness. A key challenge in both plans is that anonymizing only the subset of tuples in the final query answer may result in a privacy breach by linking of the answers from multiple queries. For example, since queries can have different sets of tuples in the final answer, a quasi-identifier attribute of a specific tuple may have 3-level generalization in Query Q1 while having a 2-level generalization in Query Q2 (that is because the set of anonymized tuples are different). This clearly results in a privacy breach since the intended level of generalization in Q1 is lost once the adversary has the results from Q2. Therefore, the anonymization should relay on the entire collection of tuples in the base table instead of considering only the final query answer. Conceptually, if a logical A-view is defined on a table T, then any query on that A-view will use either of the two query plans presented in Fig. 6. The two plans are described next.

5.1. Anonymize-then-Select Plan

The Anonymize-then-Select plan performs the anonymization as early as possible in the query pipeline, i.e., immediately after the table-scan. Consequently, the selection predicates and/or projected attributes are not pushed inside the table-scan operator but rather delayed until after the anonymization is performed on the entire table. In the Anonymize-then-Select plan, a block of tuples is read from disk and these tuples are then anonymized and pipelined to the next query operator in the plan. Thus, the anonymization operator does not block the query pipeline until all tuples in the base table are anonymized. Instead it operates as a block-level anonymization. The detailed steps are as follow.

Step 1: Index-Scan

Using a direct table scan to read the data blocks from disk may produce tuples with different order depending on the table layout on disk (recall that the anonymization operator processes one block at a time). Thus, the anonymization algorithm may generate different outputs, which may lead to a privacy breach. To overcome this issue, Anonymize-then-Select plan uses a pre-defined index on the identifier attribute to perform an index-scan over the data (See Fig. 6). Thus, the anonymized blocks are guaranteed to have the same set of tuples independent of how the data is stored on disk.

Step 2: Sensitive Attribute Generalization

Each data owner can optionally specify a generalization level for SA attributes. If the generalization level is m, then SA attributes are generalized to their m^{th} ancestor in its DGH.

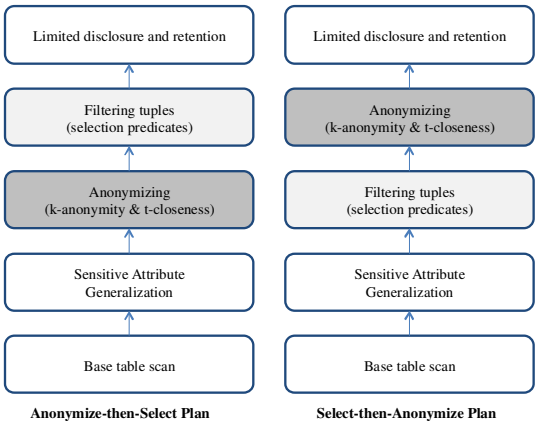


FIGURE 6: Anonymization query plans

Step 3: Block-Level Anonymization

We assume that each tuple belongs to a single data owner, e.g., patient record. If the k anonymity value is set zero, then no anonymization is required. If k =1, then it is 1-anonymous, i.e., identifiers are hidden, but will not be part of any QI group. The anonymization algorithm works as follow.

1) Retrieve the next block of tuples to process.
2) Identify the k value for each tuple from the A-view definition.
3) Mark the tuples having k = 0 as anonymized, and hide the identifier attribute of all the remaining tuples. Mark the tuples having k = 1 as anonymized .
4) Compute the probability distribution of the sensitive attribute values over the tuples in the block.
5) While there exist unanonymized tuples, repeat the following:
   5.1) Select the QI attribute having the maximum number of distinct values → Say QI_{max}.
   5.2) Generalize QI_{max} one level in its DGH. Then, merge all tuples with identical QI values in one group.
   5.3) If a group size is larger than or equals to the group's maximum k (max k for the tuples in the group), then compute the probability distribution for this QI group and calculate the t-closeness. If the group satisfies t-closeness, all tuples in that group are marked anonymized. Otherwise, repeat Step 5.1).
   5.4) If there are not enough tuples to be anonymized, then hide all identifiers, quasi identifiers, and sensitive attributes of the remaining tuples in the block. Then mark these tuples as anonymized.
6) If there are more blocks, then go to 1), otherwise exit.

Note that the anonymization operator does not wait until the anonymization of the entire block is performed to pass on the anonymized tuples to the next node in the query tree. Instead, anonymized tuples are pipelined after each iteration (Step 5 in the algorithm).

Step 4: Selection

Selection predicates on identifier attributes will not return any anonymized tuples since the identifier values are hidden in Step 3.3 above. For quasi-identifiers, it is proposed in [13, 14] that if tuple t is the only true query result, then t will not be reported in the answer to preserve the k-anonymity requirement of t's owner. In contrast, in our model, we report t along with at least k-1 other tuples that will have the same QI generalized values. Some of these reported tuples in t's QI group could be false-positive tuples, i.e., without generalization, they do not satisfy the query. This approach of including false-positive tuples preserves the anonymity requirements as well as improves the utility of the query, i.e., the number of true-positive reported tuples.

We currently support only equality predicates on QI and SA attributes. After generalizing the values in the QI and SA attributes, the equality operator needs to be extended such that an ancestor value in a

DGH should match any descendant value under that ancestor. For this purpose, we introduce the operator AVLIKE (A-view LIKE) that is used as illustrated in the following example.

Select * from PATIENT_AVIEW where zipcode = '88512';

Assuming that zipcode is a QI attribute, then the query is re-written as:

Select * from PATIENT_AVIEW where zipcode AVLIKE '88512';

Assume that two tuples t1 and t2 with zipcodes '88512', and '88513', respectively, are both generalized to '88***', then both tuples will now satisfy the query (t1 is true-positive while t2 is false-positive). This algorithm is guaranteed to return all true-positive tuples along with their QI generalized groups.

Step 5: Limited Disclosure & Retention

The projection clause is manipulated to comply with the opt-in/opt-out choices of the tuple's owners. Otherwise, the attribute value is replaced with a null value. The conditions for limited retention are also embedded within this clause. Hence, the algorithm combines the anonymization with the limited disclosure and retention to provide maximum privacy.

5.2. Select-then-Anonymize Plan

The Select-then-Anonymize plan is motivated by the fact that the former plan needs to anonymize the entire base table even if there is only one true-positive tuple. Therefore, for high-selectivity queries, it is more efficient to first find the true-positive tuples, and then construct their QI groups. In this case, the anonymization is a tuple-level anonymization that provides even better pipelining in the query plan and better response time compared to the block-level anonymization. The trick is how to form QI groups for the true-positive tuples that are still base-table dependent and not query-answer dependent. The detailed steps are as follow.

Steps 1, 2 & 3: Scan, SA Generalization, and Selection

In this step, tuples are scanned from the base table. The limitation of using pre-defined index scan (as in Anonymize-then-Select) does not apply here since the Select-then-Anonymize plan processes one tuple at a time. The SA attributes are generalized and predicates on these attributes are re-written to use AVLIKE operator instead of equality operators. If there are predicates on identifier attributes, the tuples will pass from the selection operator. But, they will be filtered in the anonymization operator (Step 4-3 below) except for tuples having k = 0.

Step 4: Anonymization

Select-then-Anonymize is based on the fact that every tuple t belongs to a particular QI group, say g(t), in the anonymized version. Thus, the true-positive tuples are first selected, and then the QI groups for each of these tuples are then retrieved from the base table. The criterion for forming the QI groups is the same as in Anonymize-then-Select plan . One trick that Select-then-Anonymize uses is that the sequence in which the QI attributes are selected for generalization (Step 3-5.1, Section V.1) can be pre-computed for a given table, e.g., an example sequence can be: zipcode, disease, disease, zipcode, …. This order can be pre-computed by estimating the number of distinct values in each of the QI attributes after each generalization step. Given that QI generalization sequence is computed, the algorithm executes as follow.

1) Scan tuples from the table being queried. Filter the tuples based on selection predicates. If the selection predicate is based on SA, use the AVLIKE operator. Indicate if the selection predicate is based on identifier or QI attributes.
2) Identify the k-requirement for each true-positive tuple.
3) If the selection predicate is based on identifiers, then report only the tuples having k = 0.
4) Hide the identifier attribute of all tuples with k ≥ 1. Mark the tuples having k = 1 as anonymized.
5) Compute the probability distribution of the sensitive attribute values in the table.
6) For every tuple t remaining after step 5, go to Step 7.
7) Initialize set g(t) to t.
8) While the size of g(t) is less than the maximum k- requirement of any tuple in g(t) or g(t) does not satisfy t-closeness repeat Steps 9-10.

9) Select the next QI attribute to be generalized from the pre-computed order → Say QI_{max}.
10) Generalize the values in QI_{max}. Retrieve form the base table all tuples with values either matching or descendant to the value in QI_{max} and add them to g(t). Go to Step 8.
11) If the selection is based on QI attributes, return the tuples in g(t) (they are the QI group of t). Otherwise, return t only.

Step 5: Limited Disclosure and Retention

This step is performed as in the Anonymize-then-Select plan.

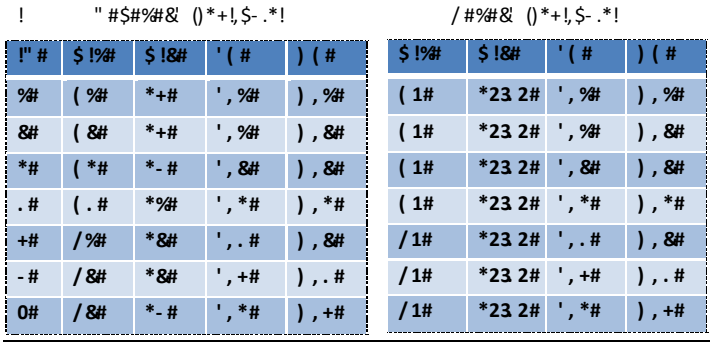


FIGURE 7: Example demonstrating privacy breach by querying QI and SA attributes

Special case: Predicates on both quasi-identifiers and sensitive attributes-

Queries that involve selection predicates on quasi-identifiers in combination with either sensitive attributes or other attributes or both need special handling. We explain this case with the help of an example given in Fig. 7. The attribute 'OA' is neither QI nor SA attributes. However, revealing the mapping between such attributes and QI attributes helps in mapping an individual to his/her SA attribute. For example, if the adversary is able to extract the information that individual with ID=1 (QI1= A1 and QI2= 35) has OA=Oa1, (s)he can easily deduce by looking at the corresponding QI group that the individual's SA value is Sa1. For this reason, consider the case of predicates on QI/SA combination separately. From Fig. 7, we observe that performing the selection before anonymization may cause privacy breach. For example, an adversary interested in finding the sensitive attribute of individual with ID=4 can issue the following queries.

```
SELECT * FROM T WHERE QI1=`A4' AND QI2=`31' AND SA=`Sa1';
SELECT * FROM T WHERE QI1=`A4' AND QI2=`31' AND SA=`Sa2';
```

Both queries will return empty results as there are no matching tuples and so the adversary successfully maps the individual to sensitive attribute value `Sa3'. It can be noted that the Anonymize-then-Select algorithm performs the selection after anonymization and so gives out some false positives even in this scenario. Therefore, Anonymize-then-Select does not suffer from this privacy breach. The issue with Select-then-Anonymize can be resolved by postponing the predicate evaluation of SA/OA attributes until after the anonymization takes place. This ensures that the results are consistent from both plans and that the adversary does not gain any further knowledge beyond what can be obtained from querying the anonymized version of the entire table.

5.3 Anonymize-then-Select vs. Select-then-Anonymize Plans

Although it is guaranteed that both plans will give query results that comply with the privacy policy, there are a few trade-offs in choosing one plan over the other.

Execution Time: Anonymize-then-Select plan has the drawback of anonymizing the entire base table even if the query is highly selective, e.g., only one true-positive tuple exists. In Contrast, Select-then-Anonymize plan anonymizes very few tuples (only the true-positives and their QI groups). However, during the anonymization of a tuple, Select-then-Anonymize plan issues several queries that may lead to a large number of random disk accesses. The number of database queries per tuple depends on the number of iterations required to form the tuple's QI group, which in turn depends on the distribution of k-

values i.e., the smaller the average k-value, the lesser the number of iterations required to form the QI group for a given tuple. Therefore, the higher the selectivity of the query and the smaller the k-values, the greater the advantage of using Select-then-Anonymize plan and vice versa.

Utility: The utility is measured in terms of the lesser number of false-positive tuples included in the answer. The utility is better for Select-then-Anonymize plan, where the maximum number of false positives for a given tuple t is |g(t)| - 1. In contrast, for Anonymize-then-Select plan, the maximum number of false positives per tuples is |Table| - 1.

In Fig. 8, we demonstrate the difference in query answers from the two plans with a simple example. The base table PATIENT consists of four attributes and five tuples. The attribute Name is an identifier, Birth and Zipcode are quasi-identifiers, and Disease is a sensitive attribute. These attributes use the domain generalization hierarchies given in Fig. 1. The issued query has a single selection predicate on Zipcode QI attribute. The table contains only one true-positive tuple as illustrated in the figure. The answers from both plans meets the privacy requirements, yet the utility of Select-then-Anonymize plan higher than that of Anonymize-then-Select since the former produces less false-positives.

6. ANONYMIZATION VIEW ON JOINED TABLES

In this section, we study anonymization over multiple tables and/or A-views. We analyse the 2-way joins (extension to more than two entities is straightforward). In Fig. 9, we classify the 2-way join cases based on the join predicates as well as whether the joins between base tables and/or A-views. Joins on QI or SA attributes are allowed using the AVLIKE operator. For example, consider joining an A-view A and a base table B. Anonymize-then-Select plan follows the regular process of constructing the A-view A, and then joins this result with the raw table B. The intuition behind not joining earlier is that the outside entity B should not be able to see the sensitive raw data associated with the A-view A. Consider the case where table B (outside entity) has only one tuple and it is joined with the A-view A on the A.ID attribute. If B were joined with the base table of the A-view A, the result could be a single tuple. Anonymizing this single tuple would not protect privacy as the mapping between the identifiers and sensitive attribute for the tuple is implicitly exposed.

Select-then-Anonymize plan also ensures that the raw table joins with the A-view only. However, as an intermediate step, Select-then-Anonymize plan calculates the semijoin of A with B. This produces the tuples in A that would be participating in the join and thus need to be anonymized. The anonymized result is then joined with the raw table. The advantage of this approach over Anonymize-then-Select plan is the higher utility in cases where join predicates are based on QI. The reasons are the same as the ones explained in Section V.

In the case of joining two A-views, Anonymize-then-Select plan follows a similar approach to the one in the previous case. Each A-view is constructed independently and the results are joined using the AVLIKE operator, if needed. Select-then-Anonymize plan calculates the semi-join of A with B and B with A, thus identifying the tuples of A and B that will participate in the join. These tuples are independently anonymized (Section V.2) and then joined.

Name	Birth	Zipcode	Disease
P1	1984	88512	Ulcer
P2	1988	88540	Indigestion
P3	1979	88541	Fever
P4	1975	89321	Fever
P5	1977	89344	Pneumonia

PATIENT

Name_op	Birth_op	Zipcode_op	K	SA Level	Purpose	Recipient
F	F	T	2	1	Treatment	Nurse
T	T	T	2	0	Treatment	Nurse
F	F	T	3	1	Treatment	Nurse
T	T	T	2	1	Treatment	Nurse
T	T	T	2	0	Treatment	Nurse

PATIENT CHOICES

QUERY: *Select * from PATIENT where zipcode='88512' purpose='Treatment' recipient='Nurse'*

TRUE POSITIVE RESULTS WITH NO PRIVACY

Name	Birth	Zipcode	Disease
P1	1984	88512	Ulcer

ALGORITHM I

Sensitive attribute generalization

Name	Birth	Zipcode	Disease
P1	1984	88512	Stomach-disease
P2	1988	88540	Indigestion
P3	1979	88541	Viral-disease
P4	1975	89321	Viral-disease
P5	1977	89344	Pneumonia

Anonymization

Name	Birth	Zipcode	Disease
*	1980-1990	88***	Stomach-disease
*	1980-1990	88***	Indigestion
*	1970-1980	****	Viral-disease
*	1970-1980	****	Viral-disease
*	1970-1980	****	Pneumonia

Selection & Limited Disclosure

Name	Birth	Zipcode	Disease
*		88***	Stomach-disease
*	1980-1990	88***	Indigestion
*		****	Viral-disease
*	1970-1980	****	Viral-disease
*	1970-1980	****	Pneumonia

ALGORITHM II

Sensitive attribute generalization

Name	Birth	Zipcode	Disease
P1	1984	88512	Stomach-disease
P2	1988	88540	Indigestion
P3	1979	88541	Viral-disease
P4	1975	89321	Viral-disease
P5	1977	89344	Pneumonia

Selection

Name	Birth	Zipcode	Disease
P1	1984	88512	Stomach-disease

Anonymization & Limited Disclosure

Name	Birth	Zipcode	Disease
*		88***	Stomach-disease
*	1980-1990	88***	Indigestion

FIGURE 8: Example demonstrating Anonymize-then-Select versus Select-then-Anonymize plans

7. MAINTENANCE OF A-VIEWS

A logical A-view work fine if the data do not change over time. If the data gets updated, and the DBMS is re-computing the A-view with each query, then the comparison of various answers may lead to a privacy breach. The incremental maintenance of K-anonymization has been studies in [8, 18]. In our model, we proposed Materialized A-views to handle insertions to existing data. For simplicity, we consider the single-k anonymity requirement. The A-view is materialized in the format shown in Fig 10, where QI groups and the corresponding tuple ids are stored.

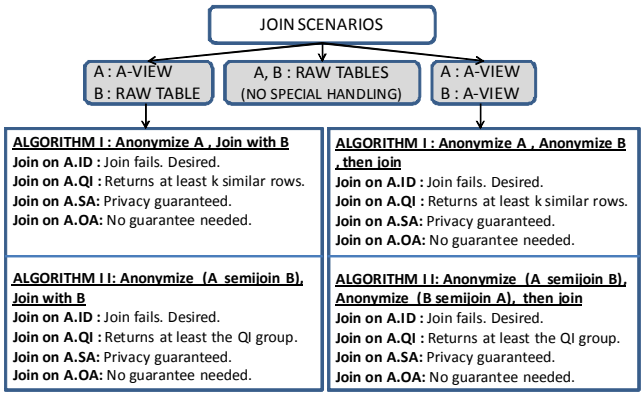


FIGURE 9: Join Scenarios

ID	QI1	QI2	SA	Others
{1,3,4}	30*	Person	{sa1,sa3,sa4}	{oa,oc,od}
{2,5}	40*	Male	{sa2,sa5}	{ob,oe}

FIGURE 10: Materialized A-view

Selection: Any query over a materialized A-view triggers a Pl/Sql procedure that uses the AVLIKE operator on QI attributes. In this case, the entire QI group is returned as output. For example, in Figure10, if the selection predicate is 'QI1 = 35', then 3 tuples formed out of the first cluster and returned as output. Selection predicates on SA or other attributes scan each of the tuples in the A-view (QI groups/ clusters) and processes the list of values to match the value in the predicate. Queries with selection predicates on identifiers return empty answer. The utility of materialized A-views is similar to that of Select-then-Anonymize plan except that in the case where the regular query result is empty. Select-then-Anonymize plan would return empty result whereas the materialized A-view may return a cluster that closely matches the predicate.

Insertion: Inserting to a base table of the materialized A-view triggers a Pl/Sql procedure that calculates the distance of the tuple being inserted from each of the tuples in the materialized A-view. Here, the distance refers to the sum of distances between corresponding quasi-identifiers. Each quasi-identifier value is a node in its DGH. The normalized distance between any two nodes in a DGH is given by the number of edges between the two nodes divided by the maximum distance between any two nodes in the DGH. This ensures that the distance measure for any QI is in the range [0 1]. The new tuple is inserted into the closest group (This may require modifying the QIs of the group). The closest group is the one where the total changes, measured in terms of normalized distance, of the inserted tuples and all existing tuples in the group, is the least. The ID, SA, and other attributes of the group are modified by adding the values of the new tuple to the corresponding lists. Upon insertions, the QI group attributes of other tuples can be only generalized to a higher node in DGH. In this case, temporal attacks from the insertion are avoided because the groups' attributes will become more generalized after the insertion. This is the trade-offs between the privacy and the utility.

Deletion: The select predicate in a delete command is processed on the base table and the IDs of the tuples to be deleted are noted. Each tuple is queried in the materialized A-view and is deleted from the group. If the group size decreases to a value less than 'k', the group is removed from the materialized A-view, and each of the remaining tuples in that group and individually inserted again into the materialized view following the insert procedure above.

Update: Update is handled as a delete of the old tuple followed by an insertion of new tuple with updated values.

8. PERFORMANCE ANALYSIS

8.1. Quality Measures

There is always a trade-off between privacy and utility in anonymization; the higher the privacy, the lower the utility of the data. The target is to maximize the utility of the anonymized data, while achieving the required levels of privacy. There are two types of utility in our system -1) utility of the query result (considers the number of false positives and false negatives) and 2) utility of the anonymized data (considers the effect of generalizing the data).

Precision and Recall: Precision is the ratio of true positives to the sum of true positives and false positives. Precision is always between 0 and 1, where 1 is the ideal value (no false positives). Since both plans return false positives when the selection predicates are on QI or SA attributes, the precision is below 1. In all other cases, the precision is 1. Recall is the ratio of the number of true positives to the sum of true positives and false negatives. Both plans are guarantee not to have false negatives (except for predicates on ID), and hence the recall is 1 in most cases.

Normalized Certainty Penalty (NCP): The NCP metric [20] measures the information loss for various choices of block sizes during the anonymization of large data. This metric can be used for the Anonymize-then-Select algorithm to identify the best block sizes to use since this algorithm is a block-level anonymization.

K-Deviation: To provide a measure of the unnecessary anonymization performed, we propose k-Deviation, which is the sum of differences between required k and achieved k for each tuple in the table. This value would be close to 0 for optimal algorithms. k-Deviation demonstrates the efficiency of the multi-k approach over the straightforward approach of anonymizing using single k, where k is set to the maximum required k over all tuples.

8.2. Experimental Results

We used a synthetically generated and real-world datasets in the experiments. The synthetic data include table 'PATIENT' with a maximum of 100K tuples, with 4 attributes {Name, Birthdate, Area and Disease}. 'Name' is the identifier, 'Birthdate' and 'Area' form the QI set, while 'Disease' is the sensitive attribute for the patient entity. Similar experiments are repeated over million tuples from the US Census Data (1990). Two identifiers 'ID' and 'NAME' are synthetically generated. The attributes 'AGE', 'SEX' and 'CITIZENSHIP' are used as QIs. 'SCHOOL (Education Level)' is considered the sensitive attribute. DGHs for QI and SA attributes are generated synthetically with a maximum depth of 4. The k-values and SA-Generalization levels for each of the million tuples are randomly generated. The k values range between 0 and 9 for all the experiments unless otherwise is specified. For all the experiments, it is assumed that the DGHs for all the attributes are in-memory data structures. The experiments are performed on Intel(R) CoreTM2 Quad CPU @ 2.83GHz machine with 3.5GB RAM.

Fig. 11 shows the runtime overhead of the anonymization operator (using Anonymize-then-Select plan) during query processing, with the block size set to 1024. Though the runtime with anonymization is asymptotically much higher than the runtime without anonymization, it is notable that the anonymization of 100k tuples is performed in less than 75 seconds for Synthetic dataset. In case of the US Census data set, the anonymization of 100k tuples is performed within 30 seconds and that of 1 million tuples is performed within 300 seconds. The time taken may vary with the algorithm used (algorithms that are more efficient in terms of utility may be slower); however, the fact that anonymization at the database engine level takes reasonable time is very promising. We observed in Fig. 12 that the run time varies with different block sizes (the table size in this experiment is set to 10K tuples). The trends show that higher block sizes marginally reduce the run time. This can be attributed to the fact that the larger numbers of tuples help in faster formation of QI groups and thus need a lesser number of iterations. Though this measurement helps in choosing the optimal block size for the dataset in hand, it is important to note that the available memory in the system (to store and anonymize the tuples) is a constraint on the maximum value of block size.

In Fig. 13, we measured the k-Deviation and NCP metric for multi-k and single-k variations. We distribute k-values in such a way that 10 percent of the tuples have high k (k=50) and the remaining values have low k (less than 5). We also show the trend for 1 percent high k values (k=50). It is observed that the k-Deviation of multi-k technique is marginally smaller than the one of the single-k technique. The reason for this is that single-k anonymization forms larger groups unnecessarily. The multi-k technique has a marginal gain over the single-k in this case of varying block sizes as well. We measured the utility loss of anonymized data for the single-k and multi-k cases and observed that NCP is close to 0 for large datasets and block sizes. This is intuitive since having large number of tuples for anonymization helps in easier formation of groups without the need for much generalization. Again, it is noted that the multi-k technique has lower NCP than the single-k case. The reason is that the multi-k technique strives for smaller group sizes (avoids unnecessary generalization) when compared to single-k, and hence results in lower k-Deviation and NCP.

In Fig. 14, we study the performance of the Select-then-Anonymize algorithm (labelled as Algorithm II in the figure). The Anonymize-then-Select algorithm is labelled as Algorithm I in the figure. We set the table size to 100K tuples and vary the selectivity of the issued queries. For queries with non-QI predicates, the response time initially increases linearly with the increase in the number of selected tuples. This is intuitive since the tuples selected based on non-QI attributes do not often tend to form QI/equivalence groups; this results in many database queries being issued for every selected tuple to find its QI group. However, the rate of increase in the running time decreases after a certain point as the algorithm comes across tuples that are already reported and thus no database queries are issued for a fraction of the selected tuples. On the other hand, the time taken by queries on QI attributes increases very slowly initially and speeds up after a certain point. The query predicates use equality or `like' operators. The reason for the this behaviour is that with a small number of selected tuples, it is highly probable that they form a part of the same QI group and thus the number of database queries to be issued is small.

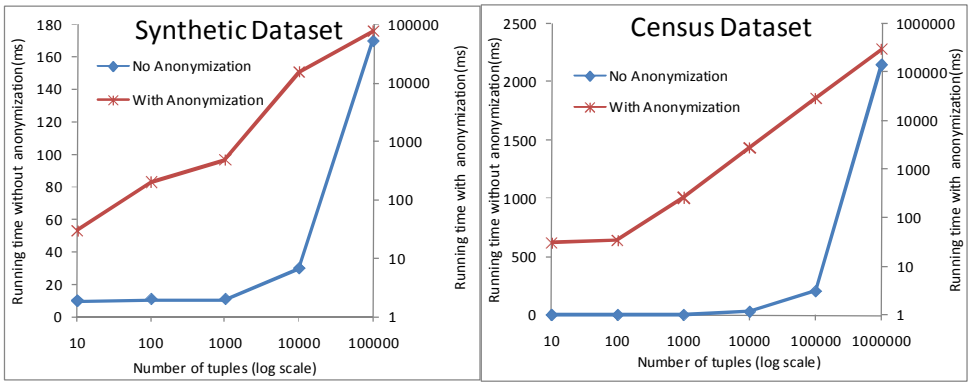


FIGURE 11: The runtime for Anonymize-then-Select algorithm under different datasets

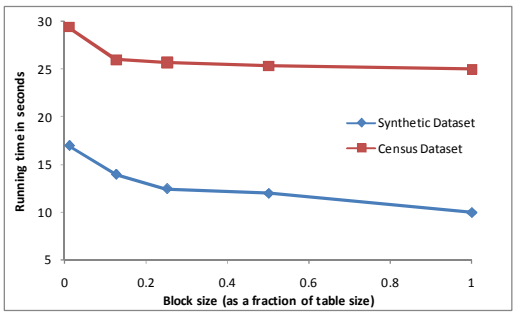


FIGURE 12: The runtime for Anonymize-then-Select under different block sizes

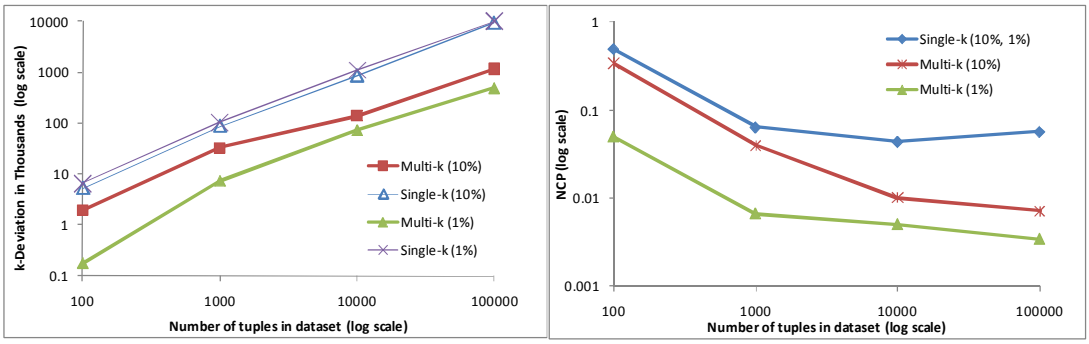


FIGURE 13: k-Deviation and NCP using Anonymize-then-Select plan (Census Dataset, sparse high-k values)

As the query result size increases, it implies that the quasi-identifiers of the selected tuples are not very similar and thus need more database queries to find their QI groups. It can be noted from Fig. 14 that Select-then-Anonymize (Algorithm II) performs better than Anonymize-then-Select (Algorithm I) if the query is highly selective, otherwise Algorithm I performs better since its anonymization cost is constant. The query selectivity and the distribution of the k values can be passed to the query optimizer to select the cheapest plan among the two. This optimization is left for future work.

In Fig. 15, we analyse the sensitivity of Select-then-Anonymize (Algorithm II) to the k value. We tested three values of k, 10, 50, and 200. The number of iterations, which maps to the number of issued database queries, increases with the increase in k. Thus, we see an increase in the runtime of Select-then-Anonymize plan, whereas Anonymize-then-Select plan takes similar runtime for all k-values.
Figs. 16 and 17 depict the performance of joins involving anonymization views. In Fig. 16, the join of an Anonymization View (100K tuples) is performed with a base table (10k tuples). As in the case of a single table, the trends are observed for QI-based and non-QI-based predicates. Fig. 17 demonstrates the runtime for the join of two anonymization views using synthetic datasets (100K and 10K tuples). we observe that the runtime almost doubled compared to joining an anonymization view with a base table. This is expected since the time taken to anonymize both relations is accounted. In both cases, the trend is similar to that in the single-table case. Similar trends are observed for Census dataset in the case Select-then-Anonymize plan, but the Synthetic data has much larger domain of distinct values for QI and non-QI attributes, and hence it displays much sharper trends.

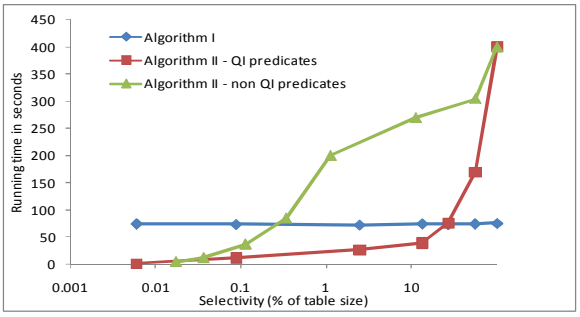


FIGURE 14: The runtime for Select-then-Anonymize plan (Synthetic Dataset)

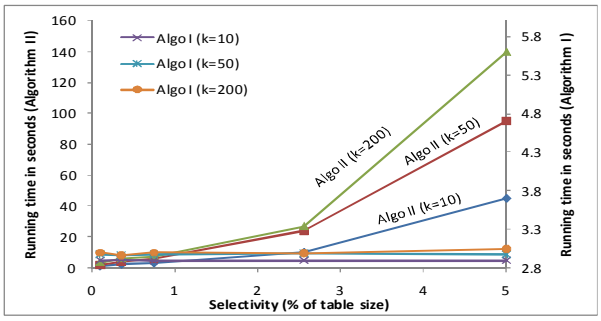


FIGURE 15: The runtime for Anonymize-then-Select plan and Select-then-Anonymize with varying `k'

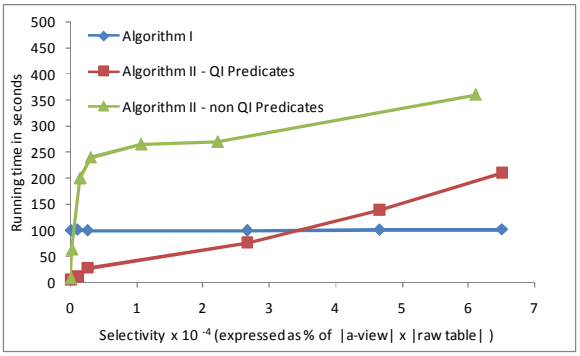


FIGURE 16: Joining an A-view and a base table

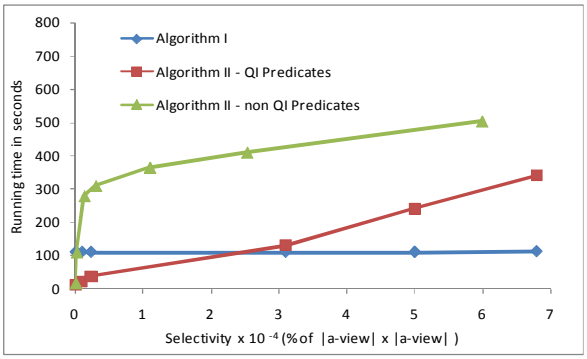


FIGURE 17: Joining two A-views

9. CONCLUSION

Protecting the privacy of the data inside the data's natural habitat, the DBMS, is an ever more demanding. It is (1) more secure, since the data is not transferred to an application layer or to a third-party, (2) more efficient, since the various query optimizations can be applied inside the database engine, (3) more flexible, since anonymization can be performed on arbitrary queries with multiple predicates, and (4) more reliable, since recovery mechanisms are already taken care of by the DBMS. In this paper, we proposed the concept of anonymization views (A-views) to enforce data privacy inside the DBMS. We presented two query plans Anonymize-then-Select and Select-then-Anonymize, which perform block-level and tuple-level anonymization, respectively, and studied the trade-offs between the two plans. For static datasets, we proposed the logical A-views to anonymize the data without the need to store the anonymized version. In contrast, for dynamically changing data, we proposed materialized A-views. Our experimental analysis and performance evaluation showed the feasibility of the proposed approach and highlighted the trade-offs among the proposed algorithms under different scenarios and settings.

REFERENCES

[1] Agarwal, R., Ameet Kini, Kristen LeFevre, Amy Wang, Yirong Xu, and Diana Zhou. Managing Healthcare Data Hippocratically. SIGMOD, 2004.

[2] Agarwal, R., Kiernan, J., Ramakrishnan Srikant, and Yirong Xu. Hippocratic Databases. VLDB. 2002.

[3] Agarwal, R., Paul, B., Grandison, T., Kiernan, J., Logan, S. and Rjaibi, W. Extending Relational Database Systems to Automatically Enforce Privacy Policies. ICDE. 2005.

[4] Agrawal, R., Bayardo, R., Faloutsos, C., Kiernan, J., Rantzau, R., and Srikant, R. Auditing Compliance with a Hippocratic Database. VLDB. 2004.

[5] Byun, J., Karma, A., Bertino, E., and Li, N. Efficient k-Anonymization using Clustering Techniques. DASFAA. 2007.

[6] Chaudhuri, S., Dutta, T. and Sudarshan, S. Fine Grained Authorization Through Predicated Grants. ICDE, 2007.

[7] Cranor, L., Langheinrich, M., Marchiori, M., Pressler-Marshall, M., and Reagle, J. The platform for privacy preferences 1.0 (P3P1.0) specification. W3C Recommendation, 2002.

[8] Jian Pei., Xu, J., Wang, Z., Wang, W., Wang, K., Maintaining K-Anonymity against Incremental Updates. 19th International Conference on Scientific and Statistical Database Management, 2007.

[9] Laura-Silva, Y N., and Aref, W. Realizing Privacy-Preserving Features in Hippocratic Databases. ICDE. 2007.

[10] LeFevre, K, Agarwal, R., Ercegovac, V., Ramakrishnan, R., Xu, Y. Limiting Disclosure in Hippocratic Databases. VLDB. 2004.

[11] Li, N., Li, T., and Venkatasubramanian, S. t-closeness: Privacy Beyond k-Anonymity and l-Diversity. ICDE. 2007.

[12] Machanavajjhala, A., Gherke, J., Kifer, D., and Venkitasubramaniam, M. l-Diversity: Privacy beyond k-Anonymity. ICDE. 2006.

[13] Nergiz, Ercan M, Clifton, C., and Nergiz, A E . Multi-relational k-Anonymity. ICDE. 2007.

[14] Qihua Wang et al. On the Correctness Criteria of Fine-Grained Access Control in Relational Databases.VLDB.'07.

[15] Sweeney, L. Achieving k-anonymity privacy protection using generalization and suppression. International Journal on Uncertainty, Fuzziness and Knowledge-based Systems, '02.

[16] Sweeney, L. Guaranteeing anonymity when sharing medical data, the datafly system. Journal of the American Medical Informatics Association, 1997.

[17] Sweeney, L. k-Anonymity:A model for protecting privacy. International Journal on Uncertainty, Fuzziness and Knowledge-based Systems, 2002.

[18] Truta, Traian Marius, and Alina Campan. K-Anonymization Incremenatal Maintenance and Optimization Techniques. Symposium on Applied Computing. 2007.

[19] Xiao, Xiaokui, and Yufei Tao. Personalized Privacy Preservation. SIGMOD, 2006.

[20] Xu, Jian, Wei Wang, Jian Pei, Xiaoyuan Wang, Baile Shi, and Ada Wai-Chee Fu. Utility-based Anonymization using Local Recoding. KDD. 2006.

[21] Padma, J., Silva, Y., Arshad, U., Aref, W. G. Hippocratic PostgreSQL. ICDE. 2009.

[22] Frank D. McSherry. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. SIGMOD, 2009

INSTRUCTIONS TO CONTRIBUTORS

Data Engineering refers to the use of data engineering techniques and methodologies in the design, development and assessment of computer systems for different computing platforms and application environments. With the proliferation of the different forms of data and its rich semantics, the need for sophisticated techniques has resulted an in-depth content processing, engineering analysis, indexing, learning, mining, searching, management, and retrieval of data.

International Journal of Data Engineering (IJDE) is a peer reviewed scientific journal for sharing and exchanging research and results to problems encountered in today's data engineering societies. IJDE especially encourage submissions that make efforts (1) to expose practitioners to the most recent research results, tools, and practices in data engineering topics; (2) to raise awareness in the research community of the data engineering problems that arise in practice; (3) to promote the exchange of data & information engineering technologies and experiences among researchers and practitioners; and (4) to identify new issues and directions for future research and development in the data & information engineering fields. IJDE is a peer review journal that targets researchers and practitioners working on data engineering and data management.

To build its International reputation, we are disseminating the publication information through Google Books, Google Scholar, Directory of Open Access Journals (DOAJ), Open J Gate, ScientificCommons, Docstoc and many more. Our International Editors are working on establishing ISI listing and a good impact factor for IJDE.

The initial efforts helped to shape the editorial policy and to sharpen the focus of the journal. Starting with volume 3, 2012, IJDE appears in more focused issues. Besides normal publications, IJDE intend to organized special issues on more focused topics. Each special issue will have a designated editor (editors) – either member of the editorial board or another recognized specialist in the respective field.

We are open to contributions, proposals for any topic as well as for editors and reviewers. We understand that it is through the effort of volunteers that CSC Journals continues to grow and flourish.

IJDE LIST OF TOPICS

The realm of International Journal of Data Engineering (IJDE) extends, but not limited, to the following:

- Approximation and Uncertainty in Databases and Pro
- Autonomic Databases
- Data Engineering
- Data Engineering Algorithms
- Data Engineering for Ubiquitous Mobile Distributed
- Data Engineering Models
- Data Integration
- Data Mining and Knowledge Discovery
- Data Ontologies
- Data Privacy and Security
- Data Query Optimization in Databases
- Data Streams and Sensor Networks
- Data Warehousing
- Database Tuning
- Database User Interfaces and Information Visualiza
- Knowledge Technologies
- Metadata Management and Semantic Interoperability
- OLAP and Data Grids
- Personalized Databases
- Query Processing in Databases
- Scientific Biomedical and Other Advanced Database
- Semantic Web
- Social Information Management
- Spatial Temporal

CALL FOR PAPERS

Volume: 3 - **Issue:** 3 - August 2012

i. Paper Submission: May 31, 2012 **ii. Author Notification:** July 15, 2012

iii. Issue Publication: August 2012

CONTACT INFORMATION

Computer Science Journals Sdn BhD
B-5-8 Plaza Mont Kiara, Mont Kiara
50480, Kuala Lumpur, MALAYSIA

Phone: 006 03 6207 1607
006 03 2782 6991

Fax: 006 03 6207 1697

Email: cscpress@cscjournals.org

CSC PUBLISHERS © 2012
COMPUTER SCIENCE JOURNALS SDN BHD
M-3-19, PLAZA DAMAS
SRI HARTAMAS
50480, KUALA LUMPUR
MALAYSIA

PHONE: 006 03 6207 1607
006 03 2782 6991

FAX: 006 03 6207 1697
EMAIL: cscpress@cscjournals.org