Gradual Clustering Algorithms

Fei Wu, Georges Gardarin

PRiSM Laboratory
University of Versailles
45 Avenue des Etats-Unis
78035 Versailles Cedex, FRANCE
Email: <Firstname>.<Lastname>@prism.uvsq.fr

Abstract

Clustering is one of the important techniques in Data Mining. The objective of clustering is to group objects into clusters such that objects within a cluster are more similar to each other than objects in different clusters. The similarity between two objects is defined by a distance function, e.g., the Euclidean distance, which satisfies the triangular inequality. Distance calculation is computationally very expensive and many algorithms have been proposed so far to solve this problem. This paper considers gradual clustering problem. From practice, we noticed that user often begin clustering on a small number of attributes, e.g., two. If the result is partially satisfying, user will continue clustering on a higher number of attributes, e.g., ten. We refer to this problem as gradual clustering problem. In fact gradual clustering can be considered as vertically incremental clustering. Approaches are proposed to solve this problem. The main idea is to reduce the number of distance calculations by using the triangle inequality. Our method first stores in an index the distances between a representative object and objects in n-dimensional space. Then these pre-computed distances are used to avoid distance calculations in (n+m)-dimensional space. Two experiments on real data sets demonstrate the added value of our approaches. The implemented algorithms are based on DBSCAN algorithm with an associated M-Tree as index tree. However the principles of our idea can well be integrated with other tree structures such as MVP-Tree, R*-Tree, etc., and with other clustering algorithms.

1 Introduction

Clustering is considered as one of the important problems in Data Mining. The objective of clustering is to partition objects into clusters such that similar objects are in the same group, while different objects are in different groups. Scalable and robust algorithms have been proposed to optimize the performance and the quality of clustering for large databases, among them BIRCH [13], CLIQUE [1], CURE [8], ROCK [9], BUBBLE [10] and others [12]. Partitional and hierarchical algorithms are generally distinguished [11]. For example, DBSCAN [5] can be seen as an improved partitional algorithm based on the density concept. To determine a cluster, DBSCAN algorithm starts from an arbitrary core point. If the neighborhood of this point satisfies a minimum density, its neighbors are in the same cluster as this point. This process is repeated for newly added neighbors until border objects are reached. Objects are stored in an index tree, e.g. M-tree or R-tree. The retrieval of neighbors of an object is realized by calling range queries on an index tree. [6] provided incremental support for DBSCAN. Their differential algorithm is able to compute efficiently new clusters when incremental insertions and deletions of objects are performed to the database.

Since no clustering algorithm is suitable for all data sets, user often like to compare the clustering qualities given by different algorithms on a small number of attributes, e.g. 2, then choose the best one to continue clustering on a higher number of attributes, e.g. 5 or 10. We refer to this phenomenon as gradual clustering problem, which is also an incremental problem but in the number of dimensions instead of objects. Noticed that horizontal and vertical enlargement can be combined together.

Distance calculation is an important operation during clustering. Our gradual clustering algorithm (GCA) aims to reduce the number of distance computations. The GCA first calculates the distances of objects in an n-dimensional space to a representative point. These distances are sorted into an ordered list with respect to their distances and are stored in a file if necessary. When doing clustering in (n+m)-dimensional space, these pre-computed distances are used to avoid some distance calculations due to the triangular inequality. Our algorithms are applied only at leaf levels of the index trees while the main idea can be easily extended to the node levels. However our purpose is to propose a new algorithm not a new index tree so that other index trees can also benefit from this technique.
We investigated two versions of our gradual clustering algorithm: one is based on one representative point, and the other is based on two representative points. Both are implemented based on DBSCAN algorithm. Our evaluations demonstrate the high efficiency of gradual clustering algorithms on two real applications: a telecommunication application and a financial data set. In summary, our gradual algorithms appear to be quite efficient and very convenient to the vertical incremental nature of clustering.

The rest of this paper is organized as follows. Section 2 describes some related work. In Section 3, we present the principle of our gradual algorithms, which perform progressive clustering on higher dimensions by using previous known information in lower dimension for filtering. Two variants are studied, one with one representative point and one with two. The experimental results are shown in Section 4. Section 5 summaries our work and points out future research directions.

2 Related Work

This section introduces background information used in this paper and discusses clustering algorithms proposed so far.

2.1 Background

The distance between two objects is calculated by a distance function, e.g., the $L_p$ norm. The $L_p$ norm is defined as $\left( \sum_{i=1}^{n} x_i^p \right)^{1/p}$. Set $p$ to 2, we get $L_2$, the Euclidean distance, which is the most used distance function in real world. Suppose $O_x$, $O_y$ and $O_z$ are three objects, the Euclidean distance satisfies the following properties:

- $d(O_x, O_y) = d(O_y, O_x)$ (symmetry)
- $d(O_x, O_y) > 0$ ($O_x \neq O_y$) and $d(O_x, O_x) = 0$ (non negativity)
- $d(O_x, O_y) \leq d(O_x, O_z) + d(O_y, O_z)$ (triangle inequality)

An important problem in clustering is how to efficiently answer range queries: given an object $o$ and a search distance $d$, we want to retrieve all objects whose distances to $o$ are less than or equal to $d$.

Up to now, many efforts have been made to tackle range queries. The FastMap algorithm [7] transforms a matrix of pairwise distances into a set of low-dimensional points. The Vantage Point (VP) Tree [3] partitions a data set according to their distances with respect to a reference (vantage) point. The median distance value is used as a separator to partition objects into two subsets. The MVP-Tree [2] extended this idea by using multiple vantage points, and exploited pre-computed distances to reduce the number of distance computations at query time. The paged balanced M-Tree [4] partitioned objects on the basis of their relative distances, as measured by a specific distance function $d$, and stores these objects into fixed-size nodes, which correspond to constrained regions of the metric space. The M-Tree reduces both the number of page reads and the number of distance computations.

The main idea behind these approaches is to partition the data space with respect to one or two vantage points. At each tree node level, points are divided into two groups according to their distances to the representative point. One group contains points whose distances to the vantage point are less than or equal to a certain radius. The other group contains points whose distances are larger than this radius. When treating range queries, the triangular inequality can be applied to filter some unsatisfying points, thus the search space is reduced.

2.2 Clustering algorithms

Recently a number of clustering algorithms for large databases has been proposed. CLARANS (Clustering Large Applications based upon RANdomized Search) proposed in [NH94] was the first spatial clustering algorithm. It applies a random search-based method to find an "optimal" clustering. The performance of CLARANS is close to quadratic in the number of points. [13] introduced their BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies) algorithm which employed the concepts of Clustering Feature and used CF tree. Clustering feature is summarizing information about a cluster, and is stored in a CF tree. The basic idea is to read objects one by one into main memory and insert them into the appropriate cluster. STING [12] is a statistical information grid approach to spatial data mining. The idea is to capture statistical information associated with spatial cells in such a manner that whole classes of queries and clustering problems can be answered without recourse to the individual objects. The CURE algorithm introduced by [8] represents each cluster by a certain fixed number of points instead of only one point and then shrinks these points toward the center of the cluster by a specified fraction. BUBBLE and BUBBLE-FM are the first scalable clustering algorithms for metric spaces [10]. BUBBLE-FM outperforms BUBBLE by reducing the number of distance calculations with the help of FastMap [7].

Different from these algorithms, DBSCAN [5] was based on the density concept. Two parameters $Eps$ and $MinPts$ are used in the algorithm to control the density of normal clusters. The algorithm begins with an arbitrary core point and retrieves its density-reachable
neighbors with respect to $Eps$ and $MinPts$. Then these points are visited and their corresponding density-reachable neighbors are retrieved. This procedure will continue recursively until border points are reached. All the points found so far form a cluster. An incremental version is presented in [6] for handling efficiently insertion and deletion of batches of tuples in a data warehouse.

Our algorithms are based on the DBSCAN algorithm with an M-tree. To make this paper self-contained, the pseudo code is given in Figure 1. The function \texttt{retrieve_neighbors}(currentObj, $Eps$) is implemented by calling the range query over the M-Tree.

![Figure 1. — Sketch of the DBSCAN algorithm](image)

### 3 Gradual Clustering with Representative Points

This section presents our two gradual clustering algorithms: one representative point (OneResPoint GCA) and two representative points (TwoResPoints GCA). The basic idea is to use the relative distances of objects so that useless distance computations can be avoided due to the triangular inequality.

#### 3.1 Gradual Clustering with One Representative Point

Since the data space is partitioned into several regions by an index tree, retrieving neighbors of a given object needs not only to visit the region that contains this given object but also to visit its neighbor regions. Consider an example given in Figure 2. Let the data space be divided into four regions A, B, C and D. To retrieve the neighbors of the point $obj$ with the radius equal to 1, all these four regions have to be visited and the distances between the points contained in these four regions and the point $obj$ will be calculated.

![Figure 2. — An example of range query](image)

![Figure 3. — Filtering with a reference point](image)

By adding a vantage point $rp$ as shown in Figure 3, the data space is further divided into 4 partitions according to a certain radius value. Suppose the radius value here is set to 1. Partition 0 contains points whose distances to $rp$ are within 1, while partition 1 contains those whose distances to $rp$ are larger than 1 but less than 2, etc. By this way, points in the same region could be distinguished from each other. For example, two points in region B can be distinguished from $obj$ since they are in partition 0, while $obj$ is in partition 1. Now let us consider the precedent range query: retrieve the neighbors of $obj$ within the radius 1. Note that the distance of two points is at least the absolute difference of the distances between these two points and another point, i.e., $|d(Ox, Oz) - d(Oy, Oz)| < d(Ox, Oy)$. Since the distance between $obj$ and $rp$ is 1, the neighbors of $obj$ must be in partition 0, 1 and 2, but not in partition 3. When determining which points in region D are possible neighbors of $obj$, it is no use to concern three gray points in partition 3. The OneResPoint GCA is based on this idea. A global vantage point is chosen in order to filter non-qualifying points at leaf levels.

The index trees proposed so far are very efficient to answer range queries. We do not think the OneResPoint algorithm is of significant computational interest when faced with only one range query. However, given $n$ objects, DBSCAN algorithm needs to submit $n$ range queries. If we can reduce on average $k$ distance computations for every object, in total $kn$ calculations could be avoided, which is not a trivial cost when $n$ is large.
Note that the distance between two objects always increases when adding dimensions since the distance is non-negative. From this fact, we derive that two objects may be neighbors in a (n+m)-dimensional space if and only if they are close enough in a lower n-dimensional space. The relation “close enough” is more precisely introduced as follows.

**Lemma 3.1:** Let $O = \{O_1, O_2, ..., O_l\}$ be a set of objects. Each object has $D_k$ dimensions, $1 \leq k \leq l$. Suppose two objects $O_i$, $O_j$ and a reference point $O$. We denote an n-dimensional space $S_n$ as $\cup_n \{D_j \mid D_j \in D\}$, with $n < l$; an $(n+m)$-dimensional space is denoted as $S_{nm} = S_n \cup_m \{D_j \mid D_j \in D and D_j \in S_n\}$ with $(n+m) \leq l$. Suppose the distances between $O_i$, $O_j$ and $O$ in the n-dimensional space $S_n$ are $d_n(O_i, O)$, $d_n(O_j, O)$ respectively. Given a threshold $Eps$ for $S_{nm}$ space, $O_i$ and $O_j$ are possible to be neighbors in $S_{nm}$ if and only if $|d_n(O_i, O) - d_n(O_j, O)| \leq Eps$. Points that satisfy this condition are said to be “close enough” in the n-dimensional space.

**Proof:**

According to the triangle inequality: $|d(O_i, O) - d(O_j, O)| \leq |d(O_i, O) + d(O_j, O)|$, the distance between two points $O_i$ and $O_j$ is at least $|d(O_i, O) - d(O_j, O)|$. If $|d(O_i, O) - d(O_j, O)|$ is larger than a certain $Eps$, $d(O_i, O_j)$ will also be larger than $Eps$. Since the distance is non-negative, the distance $d(O_i, O_j)$ in $(n+m)$-dimensional space is at least the same distance in n-dimensional space. Hence we have $d_{n+m}(O_i, O_j) = d_n(O_i, O) + d_n(O_j, O)$. For $d_{n+m}(O_i, O_j) \leq Eps$, we have $d(O_i, O_j) \leq Eps$.

The distances $d(O_i, O)$ of all objects $O_i$ are pre-computed and stored in an index with respect to their distances. This index can be saved in a file if necessary. In our implementation, the zero point is chosen to be the representative point. Let us give an example on how to applying Lemma 3.1. Consider a telecommunication table shown in Table 1. Each tuple represents a customer. Attribute AVG_LAST indicates the average lasting call time. WEEKEND represents the call time during weekend, while TOTAL indicates the total call time. Before doing clustering in 3-dimensional space, the relative distances of all these records in a 2-dimensional space (AVG_LAST, WEEKEND) are first calculated:

$\begin{align*}
\text{d}(1.0) &= \sqrt{54^2 + 152^2} = 161.1 \\
\text{d}(2.0) &= \sqrt{33^2 + 100^2} = 105.6 \\
\text{d}(3.0) &= \sqrt{46^2 + 125^2} = 133.3 \\
\end{align*}$

Suppose $Eps$ is set to 40 in a 3-dimensional space. According to the Lemma 3.1, we can say that customer1 and customer2 cannot be neighbor since $|d(1, O) - d(2, O)| = 56 > Eps$. That means they are not close enough in 2-dimensional space, thus their distance function call in 3-dimensional space is avoided.

### Table 1 — A telecommunication table

<table>
<thead>
<tr>
<th>Customer</th>
<th>AVG_LAST</th>
<th>WEEKEND</th>
<th>TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>54</td>
<td>152</td>
<td>1200</td>
</tr>
<tr>
<td>2</td>
<td>33</td>
<td>100</td>
<td>1500</td>
</tr>
<tr>
<td>3</td>
<td>46</td>
<td>125</td>
<td>1000</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

In general, the processing of range queries can be decomposed into two steps: find candidates and verify them by calling the distance function (Figure 4(a)). The OneResPoint algorithm filters the candidates based on the Lemma 3.1 before the “verify” step (Figure 4(b)). Only those points, which are close enough in the n-dimensional space, will be passed to the “verify” step. Certainly, the pruning could also be applied to internal nodes of the tree. However, we do not handle this problem in order to keep the algorithm independent of the kind of index trees. Since the OneResPoint algorithm performs the filtering only at leaf nodes, it can be integrated with other index trees.

![General Process](image1)

(a) General Process

![Our Process](image2)

(b) Our Process

**Figure 4. — Comparison of general range query process and our process**

Another problem is to choose which attributes for pre-calculation. We developed a heuristic for solving this problem. This heuristic is based on the following observation. Note that the values of an attribute form a circle $C$ with a radius $R$. This circle represents the distribution of this attribute. The radius $R$ is in fact the distance between the average value and the absolute largest value of this attribute. Larger the radius of a circle, more impact on the distance result has this attribute. Here we must point out that there are some attributes, e.g. user-id, which have no meanings for clustering, while their distributions may have a big circle. Such kind of attributes can be filtered out only by user. Let $x_i (i = 1, ..., n)$ be the values of an attribute.

Our process for choosing important attributes is composed of the following steps:

1. Normalize the values by the formula:

$$
\bar{x} = \frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{x})^2
$$

where $\bar{x}$ is the average value (i.e., $\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i$).
2. Calculate the radius $R$ by the formula:
$$R = \sqrt{\frac{\sum (x_i - \bar{x})^2}{n}}.$$ 

3. Sort the radius of all attributes. The first $m$ largest circles are the most important attributes, and will be chosen for the pre-calculation.

3.2 Gradual Clustering with Two Representative Points

The OneResPoint algorithm partitions the data space with one vantage point, while the TwoResPoints algorithm chooses two representative points for filtering. That means all objects will keep two relative distances to two reference points. In OneResPoint GCA, when two points are in the same partition divided by the zero point, but at the opposite side as shown in Figure 5, the algorithm cannot realize this situation and calculates their distance. However this calculation can be avoided in the TwoResPoints with the help of the second representative point $r_{p2}$.

![Figure 5. — Principle of the TwoResPoints GCA](image)

There are two ways to choose the second point: an extreme point or a random point. An extreme point is a point whose values are the maximum values of all attributes, i.e., $<\max(a_1), \max(a_2), \ldots, \max(a_n)>$ where $a_i$ is an attribute. We have implemented these two alternatives named TwoResPoints_Ext and TwoResPoints_Ran respectively, and will compare them in the following Section.

4 Performance Evaluation

This section reports our experimental results in comparing four algorithms: DBSCAN, OneResPoint, TwoResPoints_Ext and TwoResPoints_Ran. All these algorithms have been implemented using an M-tree index in C++. For the three new proposed algorithms, we have pre-calculated the distances between the representative points and all other objects in two dimensions. These distances are sorted and are stored in a file. All experiments start from three dimensions.

For learning the behaviors of new algorithms, two real data sets having different data distribution are used. One is a 9-dimensional telecommunication table, which contains 50,000 points. The other one is a 5-dimensional stock table, which contains 80,000 points. Their geometric spaces in two dimensions are illustrated in Figure 6 and Figure 7 respectively. The figure 6 shows that the distribution of the Stock data set is almost linear. The Tele data set in the figure 7 spans nearly a rectangular region.

![Figure 6. — Stock data](image) ![Figure 7. — Tele data](image)

The performance evaluations of the four algorithms are presented from Figure 8 to Figure 13. The number of distance computations performed by each algorithm is compared from Table 2 to Table 7. Figure 8, Figure 9, Table 2 and Table 3 show the experimental results for the Tele data set. Figure 8 and Table 2 give the results when clustering 25000 points randomly chosen from the Tele data set, while Figure 9 and Table 3 show the results on the whole Tele data set. We have randomly chosen 20000, 40000, 60000, 80000 points from the Stock data set for clustering and the results are shown from Figure 10 to Figure 13 and from Table 4 to Table 7.

Figure 8 compares the CPU cost in clustering 25000 objects from the Tele data set. The number of dimensions to add gradually increases from 1 to 7, i.e., the total number of dimensions is gradually changed from 3 to 9. Table 2 compares the number of distance calculations performed.

From Figure 8, we observe that the behaviors of these four algorithms are somehow similar. The OneResPoint algorithm outperforms the DBSCAN by a factor of almost 50%. The two TwoResPoints algorithms are both better than the OneResPoint, while in this case there is no obvious difference between the TwoResPoints_Ext and TwoResPoints_Ran algorithms. Comparing Figure 8 and Table 2, we can see that when increasing the number of dimensions, although the percentage of distance calculations avoided in larger dimensional space is less than in smaller one, the CPU cost percentage gained increases. The explanation is the tradeoff between the “close enough” test and the distance calculation. For all candidate points, the gradual algorithms will do “close enough” test, and only those qualified ones will further perform distance computation. In contrast, DBSCAN algorithm calls distance function directly for every candidate object. When the number of dimensions is small, the cost of calculating distances can be compared to the cost of the “close enough” test. However the “close enough” test is constant in the number of dimensions, while distance calculation becomes more expensive when increasing
the number of dimensions. Thus the gradual clustering algorithms need to avoid much more distance function calls in small dimensional space in order to gain CPU cost than in large dimensional space. The same phenomena can be found in Figure 9 and Table 3.

Figure 9 depicts the CPU cost comparison in clustering the whole Tele data set, while Table 3 compares the number of distance calculations of each algorithm. From this figure, we can see that the TwoResPoints Ran algorithm is close to the OneResPoint algorithm, while the TwoResPoints Ext is the best among all these four algorithms. The reason why the TwoResPoints Ext is better than the TwoResPoints Ran is that the TwoResPoints Ext partitions the data space more effectively than TwoResPoints Ran. Thus the TwoResPoints Ext avoids much more distance computations than the TwoResPoints Ran as shown in Table 3.

Figure 10 to Figure 13 depict the comparison results on the Stock data set with 20000, 40000, 60000, 80000 points by gradually adding 1, 2 and 3 dimensions. Table 4 to Table 7 show the number of distance calculations performed by each algorithm. Comparing the Figure 6 and the Figure 7, it is clear that the geometry of the Stock data set is quite different from that of the Tele data set. The Stock data set is almost linear, having no many neighbor partitions for each partition. In this case neither OneResPoint nor TwoResPoints reduce much distance computations.

In Figure 10, it appears that sometimes DBSCAN even becomes the best algorithm. The reason is that although all other three gradual algorithms reduce the number of distance computations, they need extra CPU cost to perform the close enough test. When they can not filter a sufficient number of objects, the test CPU cost is not compensated and more CPU cost is needed than DBSCAN. Since the TwoResPoints GCA needs more close enough tests than the OneResPoint, that is why sometimes it is worse than the OneResPoint algorithm. The same phenomena can be found from the Figure 11 to Figure 13 and from Table 5 to Table 7.
<table>
<thead>
<tr>
<th>Nb. Of attrib. To add</th>
<th>DBSCAN</th>
<th>OneResPoint</th>
<th>TwoResPoints_Ran</th>
<th>TwoResPoints_Ext</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5843309</td>
<td>548179</td>
<td>231374</td>
<td>228048</td>
</tr>
<tr>
<td>2</td>
<td>12162196</td>
<td>2721360</td>
<td>1519409</td>
<td>1507698</td>
</tr>
<tr>
<td>3</td>
<td>20563076</td>
<td>5316566</td>
<td>3005649</td>
<td>2973397</td>
</tr>
<tr>
<td>4</td>
<td>26413586</td>
<td>7135544</td>
<td>4137546</td>
<td>4082327</td>
</tr>
<tr>
<td>5</td>
<td>24510365</td>
<td>6906454</td>
<td>4202662</td>
<td>4160411</td>
</tr>
<tr>
<td>6</td>
<td>34019044</td>
<td>8789028</td>
<td>4971248</td>
<td>4930768</td>
</tr>
<tr>
<td>7</td>
<td>38886249</td>
<td>9921465</td>
<td>5569896</td>
<td>5508251</td>
</tr>
</tbody>
</table>

Table 2 — Number of distance computations with 25000 Tele objects

<table>
<thead>
<tr>
<th>Nb. Of attrib. To add</th>
<th>DBSCAN</th>
<th>OneResPoint</th>
<th>TwoResPoints_Ran</th>
<th>TwoResPoints_Ext</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>36325864</td>
<td>5714908</td>
<td>4382293</td>
<td>3704793</td>
</tr>
<tr>
<td>2</td>
<td>63985525</td>
<td>13382258</td>
<td>10790492</td>
<td>8086370</td>
</tr>
<tr>
<td>3</td>
<td>56484151</td>
<td>6633355</td>
<td>4622671</td>
<td>2957652</td>
</tr>
<tr>
<td>4</td>
<td>77223311</td>
<td>39735699</td>
<td>38389016</td>
<td>33861724</td>
</tr>
<tr>
<td>5</td>
<td>10710918</td>
<td>31000248</td>
<td>28860633</td>
<td>22819720</td>
</tr>
<tr>
<td>6</td>
<td>113089184</td>
<td>35495199</td>
<td>33268105</td>
<td>26742147</td>
</tr>
<tr>
<td>7</td>
<td>121669484</td>
<td>44217734</td>
<td>42043942</td>
<td>34399917</td>
</tr>
</tbody>
</table>

Table 3 — Number of distance computations with 50000 Tele objects

<table>
<thead>
<tr>
<th>Nb. Of attrib. To add</th>
<th>DBSCAN</th>
<th>OneResPoint</th>
<th>TwoResPoints_Ran</th>
<th>TwoResPoints_Ext</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2483328</td>
<td>1826575</td>
<td>1821593</td>
<td>1821568</td>
</tr>
<tr>
<td>2</td>
<td>545610</td>
<td>5333486</td>
<td>5333485</td>
<td>5333440</td>
</tr>
<tr>
<td>3</td>
<td>7833318</td>
<td>7792370</td>
<td>7792368</td>
<td>7792350</td>
</tr>
</tbody>
</table>

Table 4 — Number of distance computations with 20000 Stock objects

<table>
<thead>
<tr>
<th>Nb. Of attrib. To add</th>
<th>DBSCAN</th>
<th>OneResPoint</th>
<th>TwoResPoints_Ran</th>
<th>TwoResPoints_Ext</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>9318943</td>
<td>8506363</td>
<td>8504717</td>
<td>8504463</td>
</tr>
<tr>
<td>2</td>
<td>11957811</td>
<td>11576728</td>
<td>11574981</td>
<td>11574779</td>
</tr>
<tr>
<td>3</td>
<td>23672857</td>
<td>23599632</td>
<td>23599630</td>
<td>23599599</td>
</tr>
</tbody>
</table>

Table 5 — Number of distance computations with 40000 Stock objects

<table>
<thead>
<tr>
<th>Nb. Of attrib. To add</th>
<th>DBSCAN</th>
<th>OneResPoint</th>
<th>TwoResPoints_Ran</th>
<th>TwoResPoints_Ext</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>25858764</td>
<td>21794760</td>
<td>21763992</td>
<td>21737681</td>
</tr>
<tr>
<td>2</td>
<td>30404618</td>
<td>28960525</td>
<td>28954487</td>
<td>28953570</td>
</tr>
<tr>
<td>3</td>
<td>39321963</td>
<td>39044574</td>
<td>39044566</td>
<td>39042906</td>
</tr>
</tbody>
</table>

Table 6 — Number of distance computations with 60000 Stock objects

<table>
<thead>
<tr>
<th>Nb. Of attrib. To add</th>
<th>DBSCAN</th>
<th>OneResPoint</th>
<th>TwoResPoints_Ran</th>
<th>TwoResPoints_Ext</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>41144047</td>
<td>35307435</td>
<td>35237932</td>
<td>35229810</td>
</tr>
<tr>
<td>2</td>
<td>44064391</td>
<td>41566398</td>
<td>41556886</td>
<td>41555279</td>
</tr>
<tr>
<td>3</td>
<td>65883157</td>
<td>65524770</td>
<td>65524755</td>
<td>65524573</td>
</tr>
</tbody>
</table>

Table 7 — Number of distance computations with 80000 Stock objects
5 Conclusion

This paper presents two gradual clustering algorithms (GCAs). The cost of range query becomes an important factor of performance in clustering process, when the number of objects participating is large. We proposed gradual clustering algorithms, which benefit from clustering on some attributes to improve clustering performance on more attributes. Gradual clustering can be seen as an “incremental” clustering, but horizontally not vertically with respect to relational tables. In the studied applications, an iterative clustering on more and more dimensions is well suited and thus GCAs are often interesting.

The GCA approach pre-computes distances between a (two) global vantage point(s) and all objects in a small dimensional space, e.g., 2-dimension. Then these pre-computed distances are used to filter non-qualifying points in larger dimensional spaces e.g. 10. The OneResPoint GCA chooses one vantage point, while the TwoResPoints GCA uses two reference points to further partition the data space. From the experimental results, the gradual algorithms could outperform DBSCAN algorithm with a factor of 50%.

Future investigations are planned. As noticed, gradual clustering presents some relationships to incremental clustering. Combining the two approaches could be useful in a data warehouse environment.

6 Reference


