A Concise Representation of Range Queries

Ke Yi[1] Xiang Lian[1] Feifei Li[2] Lei Chen[1]

[1]*Dept. Computer Science and Engineering, Hong Kong U.S.T.*
Clear Water Bay, Hong Kong, China
{yike,xlian,leichen}@cse.ust.hk

[2]*Dept. Computer Science, Florida State University*
Tallahassee, FL, USA
lifeifei@cs.fsu.edu

***Abstract*—With the advance of wireless communication technology, it is quite common for people to view maps or get related services from the handheld devices, such as mobile phones and PDAs. Range queries, as one of the most commonly used tools, are often posed by the users to retrieve needful information from a spatial database. However, due to the limits of communication bandwidth and hardware power of handheld devices, displaying all the results of a range query on a handheld device is neither communication efficient nor informative to the users. This is simply because that there are often too many results returned from a range query. In view of this problem, we present a novel idea that a concise representation of a specified size for the range query results, while incurring minimal information loss, shall be computed and returned to the user. Such a concise range query not only reduces communication costs, but also offers better usability to the users, providing an opportunity for interactive exploration. The usefulness of the concise range queries is confirmed by comparing it with other possible alternatives, such as sampling and clustering. Then we propose algorithms to find a good concise representation.**

I. Introduction

Spatial databases have witnessed an increasing number of applications recently, partially due to the fast advance in the fields of mobile computing, embedded systems and the spread of the Internet. For example, it is quite common these days that people want to figure out the driving or walking directions from their handheld devices (mobile phones or PDAs). However, facing the huge amount of spatial data collected by various devices, such as sensors and satellites, and limited bandwidth and/or computing power of handheld devices, how to deliver *light* but *usable* results to the clients is a very interesting, and of course, challenging task.

Our work has the same motivation as several recent work on finding good representatives for large query answers, for example, representative skyline points in [7]. Furthermore, such requirements are not specific to spatial databases. General query processing for large relational databases and OLAP data warehouses has posed similar challenges. For example, approximate, scalable query processing has been a focal point in the recent work [6] where the goal is to provide *light, usable* representations of the query results early in the query processing stage, such that an *interactive* query process is possible. In fact, [6] argued to return concise representations of the final query results in every possible stage of a long-running query evaluation. However, the focus of [6] is on join queries in the relational database and the approximate representation is a random sample of the final query results. Soon we will see, the goal of this work is different and random sampling is not a good solution for our problem.

For our purpose, *light* refers to the fact that the representation of the query results must be small in size, and it is important for two reasons. First, the client-server bandwidth is often limited. This is especially true for mobile computing and embedded systems, which prevents the communication of query results with a large size. It is equally important for applications with PCs over the Internet. The response time is a determining factor for attracting users for using a service, as users often have alternatives, e.g., Google Map vs. Mapquest. Large query results inevitably slow down the response time and blemish the user experience. Secondly, clients' devices are often limited in both computational and memory resources. Large query results make it extremely difficult for clients to process, if not impossible. This is especially true for mobile computing and embedded systems.

Usability refers to the question of whether the user could derive meaningful knowledge from the query results. Note that more results do not necessarily imply better usability. On the contrary, too much information may do more harm than good, which is commonly known as the *information overload* problem. As a concrete example, suppose that a user issues a query to her GPS device to find restaurants in the downtown Boston area. Most readers having used a GPS device could quickly realize that the results returned in this case could be almost useless to the client for making a choice. The results (i.e., a large set of points) shown on the small screen of a handheld device may squeeze together and overlap. It is hard to differentiate them, let alone use this information! A properly sized representation of the results will actually improve usability. In addition, usability is often related to another component, namely, *query interactiveness*, that has become more and more important. Interactiveness refers to the capability of letting the user provide feedback to the server and refine the query results as he or she wishes. This is important as very often, the user would like to have a rough idea for a large region first, which provides valuable information to narrow down her query to specific regions. In the above example, it is much more meaningful to tell the user a few areas with high concentration of restaurants (possibly with additional attributes, such as Italian vs. American restaurants), so that she could further refine her query range.

A. Problem definition

Motivated by these observations this work introduces the concept of *concise range queries*, where *concise* collectively represents the *light, usable, and interactive* requirements laid out above. Formally, we represent a point set using a collection of bounding boxes and their associated counts as a concise representation of the point set.

Definition 1 Let P be a set of n points in $\mathbb{R}^2$. Let $\mathcal{P} = \{P_1, \ldots, P_k\}$ be a partitioning of the points in P into k pairwise disjoint subsets. For each subset P_i, let R_i be the minimum axis-parallel bounding box of the points in P_i. Then the collection of pairs $\mathcal{R} = \{(R_1, |P_1|), \ldots, (R_k, |P_k|)\}$ is said to be a *concise representation* of *size* k for P, with $\mathcal{P}$ as its *underlying partitioning*.

We will only return $\mathcal{R}$ as a concise representation of a point set to the user, while the underlying partitioning $\mathcal{P}$ is only used by the DBMS for computing such an $\mathcal{R}$ internally. Clearly, for fixed dimensions the amount of bytes required to represent $\mathcal{R}$ is only determined by its size k (as each box R_i could be captured with its bottom left and top right corners).

There could be many possible *concise representations* for a given point set P and a given k. Different representations could differ dramatically in terms of quality, as with $\mathcal{R}$, all points in a P_i is replaced by just a bounding box R_i and a count $|P_i|$. Intuitively, the smaller the R_i's are, the better. In addition, an R_i that contains a large number of points shall be more important than one containing few points. Thus we use the following "information loss" as the quality measure of $\mathcal{R}$.

Definition 2 For a concise representation $\mathcal{R} = \{(R_1, |P_1|), \ldots, (R_k, |P_k|)\}$ of a point set P, its *information loss* is:

$$L(\mathcal{R}) = \sum_{i=1}^{k} (R_i.\delta_x + R_i.\delta_y)|P_i|, \tag{1}$$

where $R_i.\delta_x$ and $R_i.\delta_y$ denote the x-span and y-span of R_i, respectively, and we term $R_i.\delta_x + R_i.\delta_y$ as the *extent* of R_i.

The rationale behind the above quality measure is the following. In the concise representation $\mathcal{R}$ of P, we only know that a point p is inside R_i for all $p \in P_i$. Therefore, the information loss as defined in (1) is the amount of "uncertainty" in both the x-coordinate and the y-coordinate of p, summed over all points p in P.

A very relevant problem is the k-anonymity problem from the privacy preservation domain, which observed the problem from a completely different angle. In fact, both k-anonymity and the concise representation could be viewed as clustering problems with the same objective function (1). After obtaining the partitioning $\mathcal{P}$, both of them replace all points in each subset P_i with its bounding box R_i. However, the key difference is that k-anonymity requires each cluster to contain at least k points (in order to preserve privacy) but no constraint on the number of clusters, whereas in our case the number of clusters is k while there is no constraint on cluster size. Extensive research on the k-anonymity [5], [1], [10] has demonstrated the effectiveness of using (1) as a measure of the amount of information loss by converting the point set P into $\mathcal{R}$.

Now, with Definitions 1 and 2, we define *concise range queries*.

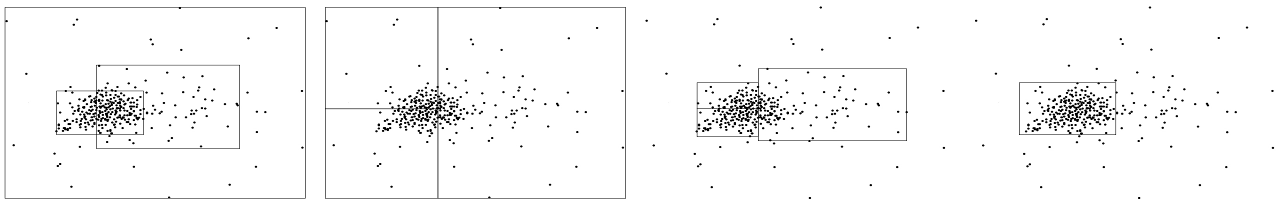
Definition 3 Given a large point set P in $\mathbb{R}^2$, a *concise range query* Q with *budget* k asks for a concise representation $\mathcal{R}$ of size k with the minimum information loss for the point set $P \cap Q$.

II. Limitation of other alternatives

Clustering techniques: There is a natural connection between the concise range query problem and the many classic clustering problems, such as k-means, k-centers, and density based clustering. In fact, our problem could be interpreted as a new clustering problem if we return the underlying partitioning $\mathcal{P}$ instead of the concise representation $\mathcal{R}$. Similarly, for existing clustering problems one could return, instead of the actual clusters, only the "shapes" of the clusters and the numbers of points in the clusters. This will deliver a small representation of the data set as well. Unfortunately, as the primary goal of all the classic clustering problems is *classification*, the various clustering techniques do not constitute good solutions for our problem. In this section, we argue why this is the case and motivate the necessity of seeking new solutions tailored specifically for our problem.

Consider the example in Figure 1, which shows a typical distribution of interesting points (such as restaurants) near a city found in a spatial database. There are a large number of points in a relatively small downtown area. The suburbs have a moderate density while the points are sparsely located in the countryside. For illustration purposes we suppose the user has a budget $k = 3$ on the concise representation.

The concise representation following our definition will partition this data set into three boxes as in Figure 1(a) (we omit the counts here). The downtown area is summarized with a small box with many points. The suburb is grouped by a larger box that overlaps with the first box (note that its associated count does not include those points contained in the first box) and all the outliers from the countryside are put into a very large box. One can verify that such a solution indeed minimizes the information loss (1). The intuition is that in order to minimize (1), we should partition the points in such a way that small boxes could have a lot of points while big boxes should contain as few as possible. If adding a new point to a cluster increases the size of its bounding box then we need to exercise extra care, as it is going to increase the "cost" of all the existing points in the cluster. In other words, the cost of each point in a cluster C is determined by the "worst" points in C. It is this property that differentiates our problem with all other clustering problems, and actually makes our definition



(a) our result. (b) k-means. (c) k-means without outliers. (d) Density based clustering.

Fig. 1. Different alternatives for defining the concise representation, $k = 3$.

an ideal choice for obtaining a good concise representation of the point set.

The result of using the modified k-means approach is shown in Figure 1(b). Here we also use the bounding box as the "shape" of the clusters. (Note that using the (center, radius) pair would be even worse.) Recall that the objective function of k-means is the sum of distance (or distance squared) of each point to its closest center. Thus in this example, this function will be dominated by the downtown points, so all the 3 centers will be put in that area, and all the bounding boxes are large. This obviously is not a good representation of the point set: It is not too different from that of, say, a uniformly distributed data set.

One may argue that the result in Figure 1(b) is due to the presence of outliers. Indeed, there has been a lot of work on outlier detection, and noise-robust clustering [2]. However, even if we assume that the outliers can be perfectly removed and hence the bounding boxes can be reduced, it still does not solve the problem of putting all three centers in the downtown (Figure 1(c)). As a result, roughly 1/3 of the downtown points are mixed together with the suburban points. Another potential problem is, what if some of the outliers are important? Although it is not necessary to pinpoint their exact locations, the user might still want to know their existence and which region they are located in. Our representation (Figure 1(a)) with $k = 3$ only tells the existence of these outliers. But as we increase k, these outliers will eventually be partitioned into a few bounding boxes, providing the user with more and more information about them.

Lastly, Figure 1(d) shows the result obtained by a density based clustering approach. A typical density based clustering, such as CLARANS [9], discovers the clusters by specifying a clustering distance ϵ. After randomly selecting a starting point for a cluster, the cluster starts to grow by inserting neighbors whose distance to some current point in the cluster is less than ϵ. This process stops when the cluster cannot grow any more. This technique, when applied to our setting, has two major problems. First, we may not find enough clusters for a given k (assume that there is a support threshold on the minimum number of points in one cluster). In this example we will always have only one cluster. Secondly, the clusters are quite sensitive to the parameter ϵ. Specifically, if we set ϵ small, then we will obtain only the downtown cluster (Figure 1(d)); if we set ϵ large, then we will obtain the cluster containing both the downtown and the suburb. Neither choice gives us a good representation of the point set.

In summary, none of the clustering technique works well for the concise range query problem since the primary goal of clustering is classification. An important consequence of this goal is that they will produce clusters that are disjoint. To the contrary, as shown in Figure 1(a), overlapping among the bounding boxes is beneficial and often necessary for our problem. Hence, we need to look for new algorithms and techniques for the concise range query problem, which consciously build the partitioning $\mathcal{P}$ to minimize the information loss.

Random sampling: Random sampling is another tempting choice, but it is easy to see that it is inferior to our result in the sense that, in order to give the user a reasonable idea on the data set, a sufficient number of samples need to drawn, especially for skewed data distributions. For example, using $k = 3$ bounding boxes roughly corresponds to taking 6 random samples. With a high concentration of points in the downtown area, it is very likely that all 6 samples are drawn from there.

Indeed, random sampling is a very general solution that can be applied on any type of queries. In fact, the seminal work of [6] proposed to use a random sample as an approximate representation of the results of a join, and designed nontrivial algorithms to compute such a random sample at the early stages of the query execution process. The fundamental difference between their work and ours is that the results returned by a range query in a spatial database are strongly correlated by the underlying geometry. For instance, if two points p and q are returned, then all the points in the database that lie inside the bounding box of p and q must also be returned. Such a property does not exist in the query results of a join. Thus, it is difficult to devise more effective approximate representations for the results of joins than random sampling. On the other hand, due to the nice geometric and distributional properties exhibited by the range query results, it is possible to design much more effective means to represent them concisely. Our work is exactly trying to exploit these nice spatial properties, and design more effective and efficient techniques tailored for range queries.

III. The Algorithms

In this section, we focus on the problem of finding a concise representation for a point set P with the minimum information loss. First in Section III-A, we show that in one dimension, a simple dynamic programming algorithm finds the optimal solution in polynomial time. Then we extend the algorithm to higher dimensions.

A. Optimal solution in one dimension

We first give a dynamic programming algorithm for computing the optimal concise representation for a set of points P lying on a line. Let $p_1, \ldots, p_n$ be the points of P in sorted order. Let $\mathcal{P}_{i,j}$ represent the optimal partitioning underlying the best concise representation, i.e., with the minimum information loss, for the first i points of size j, $i \geq j$. The optimal solution is simply the concise representation for $\mathcal{P}_{n,k}$, and $\mathcal{P}_{n,k}$ could be found using a dynamic programming approach. The key observation is that in one dimension, the optimal partitioning always contains segments that do not overlap, i.e., we should always create a group with consecutive points without any point from another group in-between. Formally, we have

Lemma 1 *$\mathcal{P}_{i,j}$ for $i \leq n, j \leq k$ and $i \geq j$ assigns $p_1, \ldots, p_i$ into j non-overlapping groups and each group contains all consecutive points covered by its extent.*

Proof: We prove by contradiction. Suppose this is not the case and $\mathcal{P}_{i,j}$ contains two groups P_1 and P_2 that overlap in their extents as illustrated in Figure 2. Let $P_i.x_l$ and $P_i.x_r$ denote the leftmost and rightmost points in P_i. Without loss of generality we assume $P_1.x_l \leq P_2.x_l$. Since P_1 intersects P_2, we have $P_2.x_l \leq P_1.x_r$. If we simply exchange the membership of $P_1.x_r$ and $P_2.x_l$ to get P_1' and P_2', it is not hard to see that both groups' extents shrink and the numbers of points stay the same. This contradicts with the assumption that $\mathcal{P}_{i,j}$ is the optimal partitioning. ■

Thus, $\mathcal{P}_{i,j}$ is the partitioning with the smallest information loss from the following $i-j+1$ choices: $(\mathcal{P}_{i-1,j-1}, \{p_i\})$, $(\mathcal{P}_{i-2,j-1}, \{p_{i-1}, p_i\})$, …,$(\mathcal{P}_{j-1,j-1}, \{p_j, \ldots, p_i\})\}$. Letting $L(\mathcal{P}_{i,j})$ be the information loss of $\mathcal{P}_{i,j}$, the following dynamic programming formulation becomes immediate.

$$L(\mathcal{P}_{i,j}) = \min_{1 \leq \ell \leq i-j+1} (L(\mathcal{P}_{i-\ell,j-1}) + \ell \cdot |p_i - p_{i-\ell+1}|), \quad (2)$$

for $1 \leq i \leq n, 2 \leq j \leq k$ and $j \leq i$. The base case is $\mathcal{P}_{i,1} = \{\{p_1, \ldots, p_i\}\}$ for $1 \leq i \leq n$. Since computing each $L(\mathcal{P}_{i,j})$ takes $O(n)$ time, the total running time is of this algorithm $O(kn^2)$.

Theorem 1 *In one dimension, the concise representation with the minimum information loss for a set of points P can be found in $O(kn^2)$ time.*

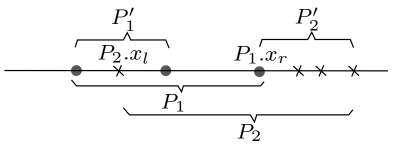


Fig. 2. Proof of Lemma 1.

B. Heuristics for two or more dimensions

Due to space limitations, we will refer interested readers to [11] for detailed descriptions of our solutions in two and higher dimensions.

IV. Related Work

The motivation of this work is very similar to the recent work of Jermaine et al. [6]. The focus of [6] is to produce approximate results for long-running join queries in a relational database engine at early stages of the query execution process. The "approximation" defined there is a random sample of the final results. As we elaborated in Section II, due to the nice geometric properties of range queries in spatial databases, it is important to design more effective and efficient methods than random sampling. The goal of this work is thus to derive such a concise representation for range queries with the minimal amount of information loss. With similar arguments, our work also bears the same motivation as finding the representative skyline points [7], however, we focus on range queries rather than dominance points.

Section II has pointed out the close relationship between the concise representation problem and classic clustering problems. I/O-efficient clustering algorithms have been studied in [12], [4]. In particular, k-medoids (k-means with the constraint that the cluster center must be a point from the input data set) and k-centers have been extended to work for disk-based data sets using R-trees [8]. Our work focuses on a completely different definition of clustering, as Section II has illustrated the limitations of using either k-means or k-centers for our problem.

Acknowledgment: Ke Yi is supported in part by Hong Kong Direct Allocation Grant (DAG07/08).

References

[1] G. Aggarwal, T. Feder, K. Kenthapadi, S. Khuller, R. Panigrahy, D. Thomas, and A. Zhu. Achieving anonymity via clustering. In *PODS*, 2006.
[2] C. Böhm, C. Faloutsos, J.-Y. Pan, and C. Plant. RIC: Parameter-free noise-robust clustering. *TKDD*, 1(3), 2007.
[3] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD*, 1996.
[4] V. Ganti, R. Ramakrishnan, J. Gehrke, and A. Powell. Clustering large datasets in arbitrary metric spaces. In *ICDE*, 1999.
[5] G. Ghinita, P. Karras, P. Kalnis, and N. Mamoulis. Fast data anonymization with low information loss. In *VLDB*, 2007.
[6] C. Jermaine, S. Arumugam, A. Pol, and A. Dobra. Scalable approximate query processing with the dbo engine. In *SIGMOD*, 2007.
[7] X. Lin, Y. Yuan, Q. Zhang, and Y. Zhang. Selecting stars: The k most representative skyline operator. In *ICDE*, 2007.
[8] K. Mouratidis, D. Papadias, and S. Papadimitriou. Tree-based partition querying: a methodology for computing medoids in large spatial datasets. *VLDB J.*, to appear, 2008.
[9] R. T. Ng and J. Han. Efficient and effective clustering methods for spatial data mining. In *VLDB*, 1994.
[10] J. Xu, W. Wang, J. Pei, X. Wang, B. Shi, and A. W.-C. Fu. Utility-based anonymization using local recoding. In *SIGKDD*, 2006.
[11] K. Yi, X. Lian, F. Li, and L. Chen. Concise range queries. Technical report, HKUST, 2008.
[12] T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: an efficient data clustering method for very large databases. In *SIGMOD*, 1996.