CONCISE RANGE QUERIES

Srikanth S.R., Madialagan M.K.

[1]Sri Venkatesa Perumal College of Engineering and Technology
Email: [1]srikanth310585@gmail.com, [2]madhivel@gmail.com

Abstract

As 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. Here range queries, as one of the most commonly used tools, are often posed by the users to retrieve needful information from a spatial database. In view of this problem, we present a 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. Unfortunately, we prove that finding the optimal representation with minimum information loss is an NP-hard problem. Therefore, we propose several effective and nontrivial algorithms to find a good approximate result. Extensive experiments on real-world data have demonstrated the effectiveness and efficiency of the proposed techniques.

Keywords —*Spatial databases, range queries, algorithms.*

I. INTRODUCTION

Spatial databases have witnessed an increasing number of applications recently, partially due to the fast advance in the fields of mobile computing and 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 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 works on finding good representatives for large query answers. General query processing for large relational databases and OLAP data warehouses has posed similar challenges. Query result in every possible stage of a long running query evaluation.

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 three reasons.

First of all, 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. This is especially important when the query results have large scale.

Second, 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.

Third, when the query result size is large, it puts a computational and I/O burden on the server. The database indexing community has devoted a lot of effort in designing various efficient index structures to speed up query processing, but the result size imposes an inherent lower bound on the query processing cost.

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. The results (i.e., a large set of points) shown on and overlap. It is hard to differentiate them, let alone use Query Q with budget k asks for a concise representation R of this information. 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.

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 Pbea set ofnpointsinIR. Let $P = \{ P1, \ldots, Pk \}g$ be a partitioning of the points in P into k pair wise disjoint subsets. For each subset Pi, let Ri be the minimum axis-parallel bounding box of the points in Pi. Then, the collection of pairs $R = \{ (R1, \backslash P1 \backslash), \ldots, (Rk, \backslash Pk \backslash) \}$ is said to be a concise representation of size k for P, with P as its underlying partitioning.

We will only return R as a concise representation of a point set to the user, while the underlying partitioning P is only used by the DBMS for computing such an R internally. Clearly, for fixed dimensions, the amount of bytes required to represent R is only determined by its size k.

Definition 2: For a concise representation $R = \{ R1, \backslash P1 \backslash), \ldots, (Rk, \backslash Pk \backslash) \}$ of a point set P, its information loss is:

$$L(R) = \sum_{i=1}^{l} Ri \cdot \dot{o}\, x + Ri \cdot \dot{o}\, y \,|\pi|$$

where $Ri, \dot{o}\, x$ and $Ri : \dot{o}\, y$ denote the x-span and y-span of Ri, respectively, and we term $Ri, \dot{o}\, x + Ri, \dot{o}\, y$ as the extent of Ri.

Definition 3: Given a large point set P in IR, a concise range Queru Q wotj bidget k asks for a concise representation R of Size k with the minimum information loss for the point set $P \cap Q$.

We can also fix the information loss L and seek for a concise representation with a minimum k (the size of the output),

Definition 4: Given a large point set P in IR, a complement concise range query Q with budget. L asks for a concise representation R of a minimum size, i.e., minimum k, for the point set $P \cap Q$ with information loss $L(R) \leq L$.

We will focus on Definition 3 and discuss the proposed solutions to complement concise range queries.

B. Summary of Contributions

The goal of a concise range query is to find a Concise representation, with the user-specified size, for all the points inside the query range. We first give a dynamic programming algorithm that finds the optimal solution in one dimension in Section 3.1. This optimization problem in two or more dimensions is NP-hard. In Section 3.2, we present a nontrivial reduction from PLANAR 3-SAT to the concise representation problem and prove its NP-hardness.

Thus, in Section 3.3, we focus on designing efficient yet effective algorithms that find good (but not optimal) concise representations.

Then, in Section 4, we explore how to speed up query processing by using an existing R-tree built on the data set P. We present an adaptive R-tree traversal algorithm that is much more efficient than answering the query exactly, and also produces high-quality concise representations of the query results. We discuss some extensions of concise range queries in Section 5. In Section 6, we demonstrate the effectiveness and efficiency of the general solution that can be applied on any type of queries.

proposed techniques with extensive experiments on real data sets. A survey of related works appears in Section 7 before concluding the paper.

II. LIMITATION OF OTHER ALTERNATIVES

A. 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. 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. Consider the example in Fig. 1, which shows a typical distribution of interesting points (such as restaurants) near a city found in a spatial database. we suppose the user has a budget $k = 3$ on the concise representation. The result of using the modified k-means approach is shown in Fig. 1b. Here, we also use the bounding box as the "shape" of the clusters. Thus, in this example, this function will be dominated by the downtown points.

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. In this example, we will always have only one cluster. Second, the clusters are quite sensitive to the parameter. Finally, we omit the result from k-centers. Hence, we need to look for new algorithms to the concise range query problem.

B. Histogram:

Our work is also related to histogram construction. Specifically, a histogram consists of several buckets, each of which stores the frequency of data points in it. The histogram has been widely used as a tool for selectivity estimation.

C. 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 be drawn, especially for skewed data distributions. Indeed, random sampling is a very Our work is exactly trying to exploit these nice spatial properties, and design more effective and efficient techniques tailored for range queries.

III. THE BASE ALGORITHMS

In this section, we focus on the problem of finding a concise representation for a point set *P* with minimum information loss. First in Section 3.1, we show that in one dimension, NP-hard in two dimensions as we show in the Section 3.2. In Section 3.3 for two or higher dimensions.

A Optimal Solution in One Dimension:

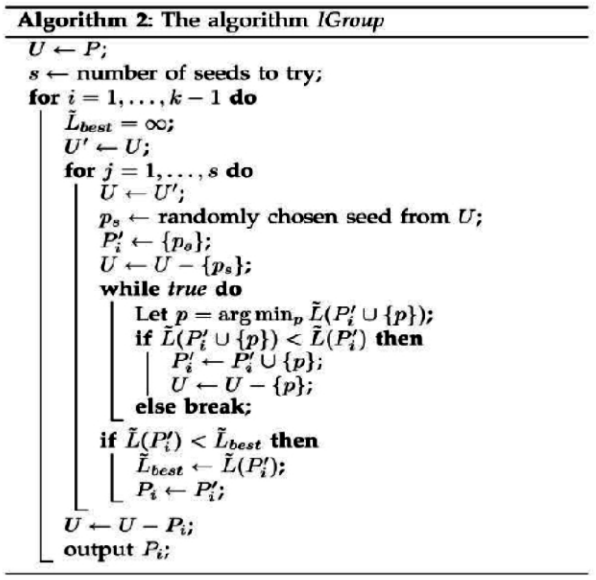
We first give a dynamic programming algorithm for computing the optimal concise representation for a set of points *P* lying on a line. Let $P1, \ldots, Pn$ be the points of *P* in sorted order. Let Pi, j represent the optimal partitioning underlying the best concise representation

Lemma 1: Pi, j for $i = n; j \leq k$ and $i = j$ assigns $P_a, \ldots P\, i$ into *j* no 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 $Pi; j$ contains two groups $P1$ and $P2$ that overlap in their extents as illustrated.

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.

B. Hardness of the Problem in 2D:

Not surprisingly, like many other clustering problems, for a point set *P* in *IR*, the problem of finding a concise representation of size *k* with minimum information loss is NP-hard. In the classical 3-SAT problem, there are *n* Boolean variables $X1, \ldots, Xn$ and *m* clauses $C1, \ldots, Cm$, where each clause is a disjunction of three variables or their negations.



```
Algorithm 2: The algorithm IGroup
U ← P;
s ← number of seeds to try;
for $i = 1, \ldots, k-1$ do
    $\tilde{L}_{best} = \infty$;
    $U' \leftarrow U$;
    for $j = 1, \ldots, s$ do
        $U \leftarrow U'$;
        $p_s \leftarrow$ randomly chosen seed from $U$;
        $P'_i \leftarrow \{p_s\}$;
        $U \leftarrow U - \{p_s\}$;
        while true do
            Let $p = \arg\min_p \tilde{L}(P'_i \cup \{p\})$;
            if $\tilde{L}(P'_i \cup \{p\}) < \tilde{L}(P'_i)$ then
                $P'_i \leftarrow P'_i \cup \{p\}$;
                $U \leftarrow U - \{p\}$;
            else break;
        if $\tilde{L}(P'_i) < \tilde{L}_{best}$ then
            $\tilde{L}_{best} \leftarrow \tilde{L}(P'_i)$;
            $P_i \leftarrow P'_i$;
    $U \leftarrow U - P_i$;
    output $P_i$;
```

Fig. 1. IGroup Algorithm

First, one can verify that the constructed point set P has The following properties:

Property 1: The l1 distance between any two consecutive points in any chain is 0.1, while the distance between any other pair of points is >0:1.

Property 2: The bounding box for a clause point and the two contacting points from one of its joining chains have extent 0.19.

Property 3: The extent of the bounding box is ≥ 0.24 for any four. Points, ≥ 0.32 for any five points, ≥ 0.36 for any six points, and $= 0.38$ for any seven points.

Lemma 2: $L\,(R\,opt) = 0.2k, 0.37\,m$, and the lower bound can be attained only if each clause point is grouped together with the two contacting points from one of its joining chains, while each of the remaining points in *P* is grouped together with one of its adjacent neighbors in its chain.

Proof: which can be found on the Computer Society Digital library at http://doi.ieeecomputersociety.org/10.1109/TKDE.2010.35.

Lemma 3: The PLANAR 3-SAT instance has a satisfying assignment if and only if $L(Ropt) = 0.2k + 0.37$ m

Proof: which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TKDE.2010.35.

The following hardness results is an immediate consequence of Lemma 3:

Theorem 2: Given a point set $PC\, IR^2$ and an integer k, the problem of finding a concise representation R of size k for P with the minimum information loss is NP-hard.

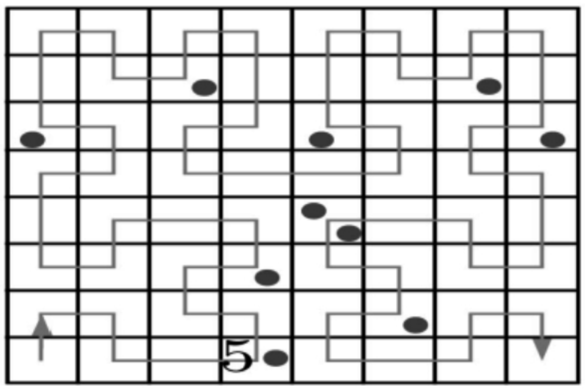


Fig. 2. The third-order Hilbert curve in two dimensions.

C. Heuristics for Two or More Dimensions:

Thus, in this section, we try to design efficient heuristic algorithms that produce an R with low information loss, although not minimum. Since our problem is also a clustering problem, it is tempting to use some popular clustering heuristic, such as the well-known k-means algorithm, for our problem as well as shown in Fig. 2.

1. Algorithm HGroup:

Given the optimal algorithm in one dimension, a straight-forward idea is to use a function $IR^2 \rightarrow IR$ to map the points of P from higher dimensions down to one dimension. Our Hilbert-curve based algorithm, called H Group, is shown in Algorithm 1.

Algorithm 1: The algorithm HGroup more direct algorithm in Compute the Hilbert value $h(Pi)$ for each point $Pi \in P$; Sort P by $h(Pi)$ and map it to one dimension; Find the partitioning P using dynamic programming; Build the concise representation R for P and return;

2. Algorithm IGroup:

More direct algorithm in two or more dimensions. It is an iterative algorithm that finds the k groups $P1; \ldots; Pk$, one at a time. We call this algorithm IGroup. We give the details of the complete algorithm IGroup in Algorithm 2. as shown in Fig. 1. There are $k-1$ iterations in the algorithm. In each iteration, we check each of the n points and choose the best one to add to the current group. In the worst case, we could check all the points $O(n)$ times. Each iteration needs to be repeated for s times with s randomly chosen seeds. So the worst case running time of IGroup is $O(skn)$.

IV. QUERY PROCESSING WITH R-TREES

In order to use the algorithms of Section 3.3 to answer a concise range query Q with budget k from the client, the database server would first need to evaluate the query as if it were a standard range query using some spatial index built on the point set P, typically an R-tree.

The algorithms presented in this section in general work with any space partitioning index structure, for concreteness, we will proceed with the R-tree as shown in Fig. 3.

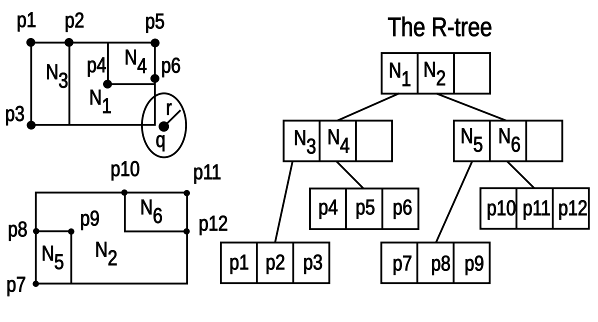


Fig. 3. R-tree

A. Basic Ideas:

For brevity we will also say a point is the MBR of itself. The basic idea of our R-tree-based algorithms is to evaluate Q in a way similar to a standard range query. However, on the other hand, since the primary goal of the R-tree is fast query processing, the MBRs do not constitute a good concise representation of the query result. overlapping among the $Ri\,s$ is often necessary and beneficial for reducing the information loss.

B. Algorithm R-BFS:

The straightforward way to find k such MBRs is to visit the part of the R-tree inside Q in a BFS manner, until we reach a level where there are at least $\alpha\, k$ MBRs. In particular, for any node u whose MBR is complete inside Q, while the associated count is estimated as assuming uniform distribution of the points in MBR(u) and Q, while the associated count is estimated as shown in equ 1.

$$\eta_{uu} \cdot \frac{\text{Area (MBR (u))} \cap \text{Q)}}{\text{Area (Q)}} \qquad \text{... (1)}$$

C. Algorithm R-Adaptive

The above BFS traversal treats all nodes alike in the R-tree and will always stop at a single level. But, intuitively, we should go deeper into regions that are more "interesting,"

i.e., regions deserving more user attention. In the algorithm R-Adaptive, we start from the root of the R-tree with an initial budget of $k = \alpha\, k$, and traverse the tree top-down recursively. It now remains to specify how we allocate the budget into $K1, \ldots, Kb$. The procedure of the recursive call visit(u, k) is outlined in algorithm 3 as shown as fig 4.

```
Algorithm 3: Recursive call visit(u, κ)
Let u_1, ..., u_b be u's children whose MBRs are
inside or partially inside Q;
Let n_i = number of points inside MBR(u_i)∩Q;
if b ≥ κ then
    output MBR(u_i)∩Q with n_i for all i;
    return;
Let A_i = Area(MBR(u_i)∩Q);
Compute κ_i as in (9) for all i = 1, ..., b;
for i = 1, ..., b do
    if κ_i = 1 then
        output MBR(u_i)∩Q with n_i;
    else
        visit(u_i, κ_i);
```

Fig. 4. Recursive call

D. The Weighted Versions of the Base Algorithms:

Our R-tree-based algorithms generate a number of MBRs, associated with counts, and pass them to the base algorithms of Section 3.3. As described, those algorithms can only process a point set. But, it is not difficult to adapt both HGroup and IGroup to handle a set of MBRs associated with counts (weights).

E. Discussions on the Shape of Query Region:

We always assume that the query region Q is an axis-parallel rectangle. In the case where the query region $Qq \cdot R-$BFS and R-Adaptive, with minor modifications has other shapes. By other estimation techniques such as histograms or sampling. Similarly, for R-Adaptive, we also need to consider the intersection between MBR(u) and Q of any shape.

F. Discussions on Other Spatial Data Types:

We now discuss how to extend our solutions of answering concise range queries on data points to that on other spatial data types such as lines or rectangles. Our extension of concise range queries on lines or rectangles can be as follows: First, we bound lines/rectangles with minimum bounding rectangles (MBRs). Then, we index the resulting MBRs in an R-tree structure by using the standard "insert" operator. For any specified concise range Q, we can conduct the concise range query on such MBRs via R-tree in the same way as that over data points.

V. EXTENSIONS

A. Supporting Attributes:

As we emphasized in Section 1, it is often useful if we can associate multiple counts with each bounding box, each of which represents the number of points with a particular attribute value, for example, Italian, Chinese, and American restaurants are referred in fig 5.

B. Complement Concise Range Queries:

As we mentioned in Section 1, the user may also be interested in asking a complement concise range query. The larger k is, the smaller the information loss is. Thus, we can conduct a binary search on *k* to find the minimum *k* that satisfies the user's requirement. This is not surprising, since the complement concise range query problem can be also shown to be NP-hard following our proof for the primary problem.

C. Progressive Refinement:

Another useful yet simple extension of our algorithms is to support progressive refinement of the concise representation. In this case, the user with a slow network connection does not specify *a k* before hand.

VI. EXPERIMENTS

A. Experimental Setup:

We have implemented our two base algorithms, HGroup and IGroup. Specifically, we used the R*-tree to index all the points in the data set. R-BFS + IGroup, R-BFS + HGroup, R-Adaptive + IGroup, and Radaptive + HGroup.

B. Data Sets:

We tested the query performance of our approaches over three real data sets, road networks from North America (NA), California (CA), and City of San Joaquin County (TG), and one synthetic data set, Skew. Specifically, for real data sets, NA and CA are from digital chart of the world server, whereas TG is from. The figures 6 and 7 shown the experimental results of CA, NA and TG.

C. Visualization of Results and Interactive Exploration:

We first did some test queries to see if the concise representation indeed gives the user some intuitive high-level ideas about the query results. Starting from here, the user can interactively narrow her query down to areas of her interests, such as high-density areas or areas with medium density but closer to locations of interest

D. Experimental Results with Different Approaches:

We now compare six approaches R-BFS IGroup, R-Adaptive+ IGroup, R-BFS +HGroup, R-Adaptive+ HGroup, k-means, and MinSkew.

*E. Experimental Results with Varying **k***

Next, we started to look into the performance of these six algorithms. In the first set of experiments, we fixed the query size at $0.1 \times 0:1, \alpha = 8$, and varied *k* from 20 to 100. Figs. 10a, 10b, and 10c present experimental results on synthetic Skew data set, comparing the six approaches, we will only report the results with our four algorithms R-BFS + IGroup, R-Adaptive+IGroup, R-BFS + HGroup, and *R* Adaptive + HGroup. We next investigate the trend of our approaches for different *k* values on CA data set in Fig. 9. The show data set is shown in figures 8. 11, 13 and 14.

F. Experimental Results with Varying α

In the second set of experiments, we fixed the query size at 0.1×0.1, $k = 60$, and varied a from 2 to

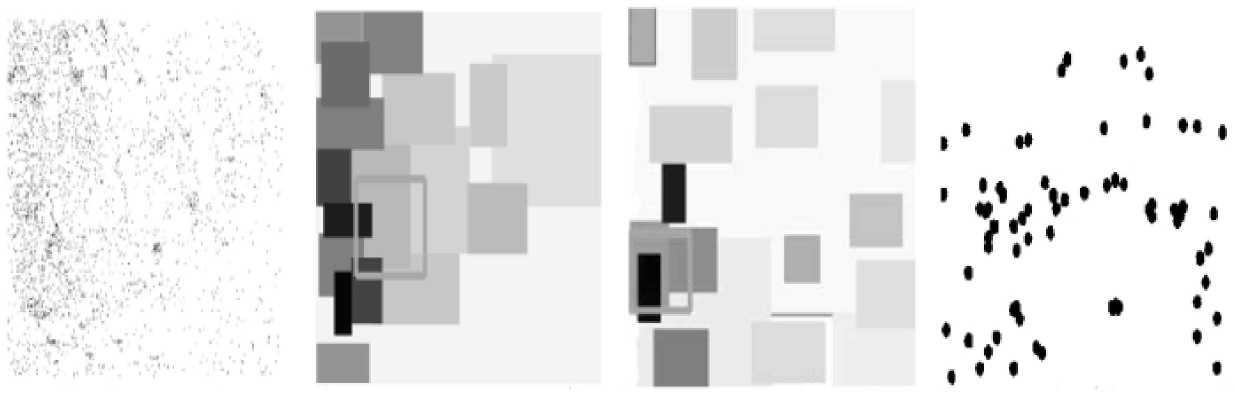
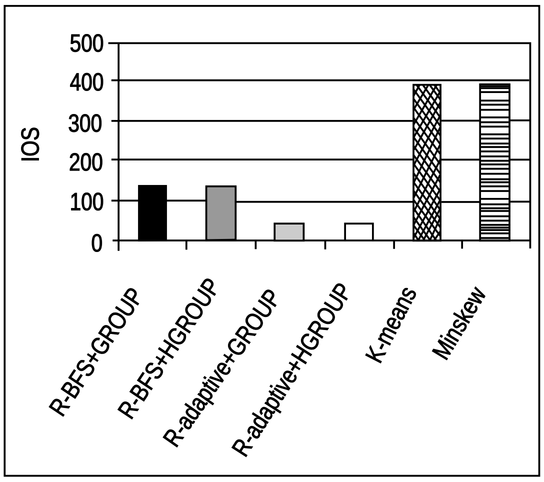
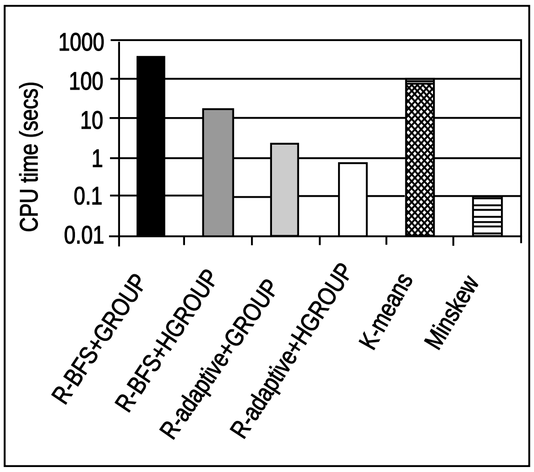

(a) (b) (c) (d)

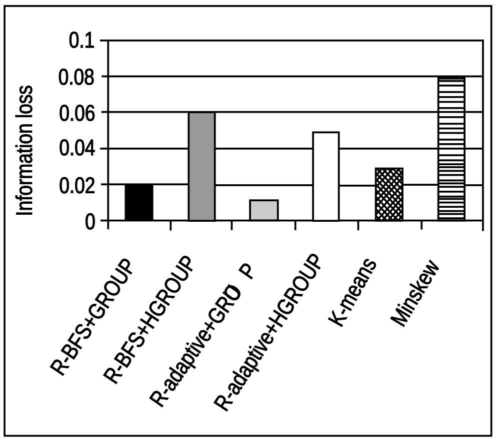
Fig. 5. Supporting Attributes



(a)

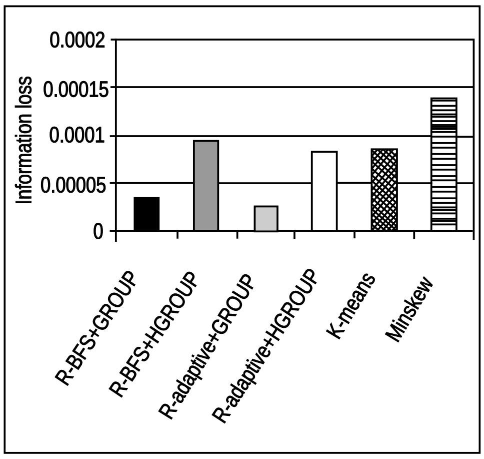


(b)

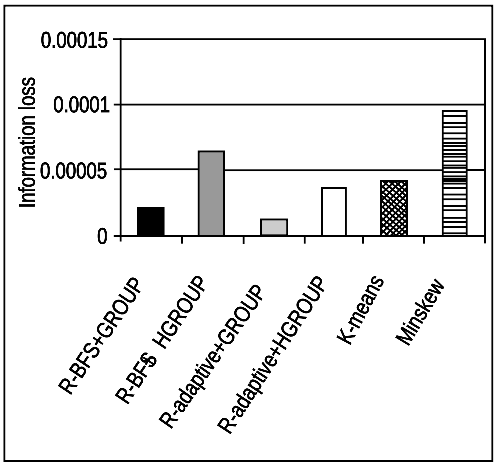


(c)

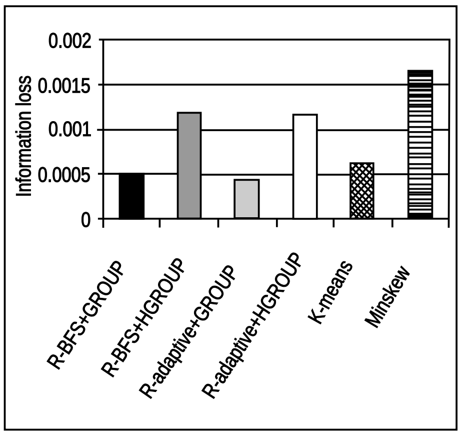
Fig. 6. Experimental results with different approaches on the CA data set. (a) I/O cost. (b) CPU time. (c) Information loss.



(a)

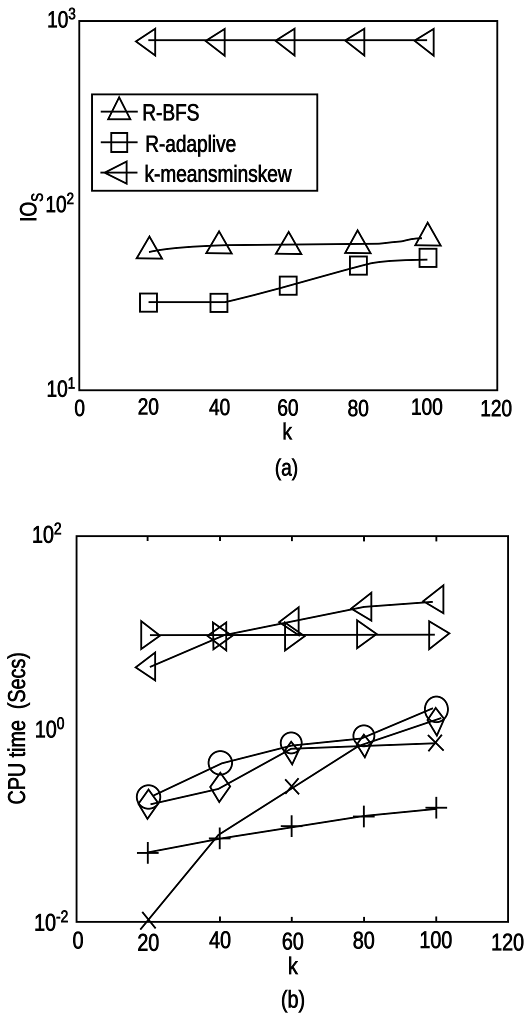


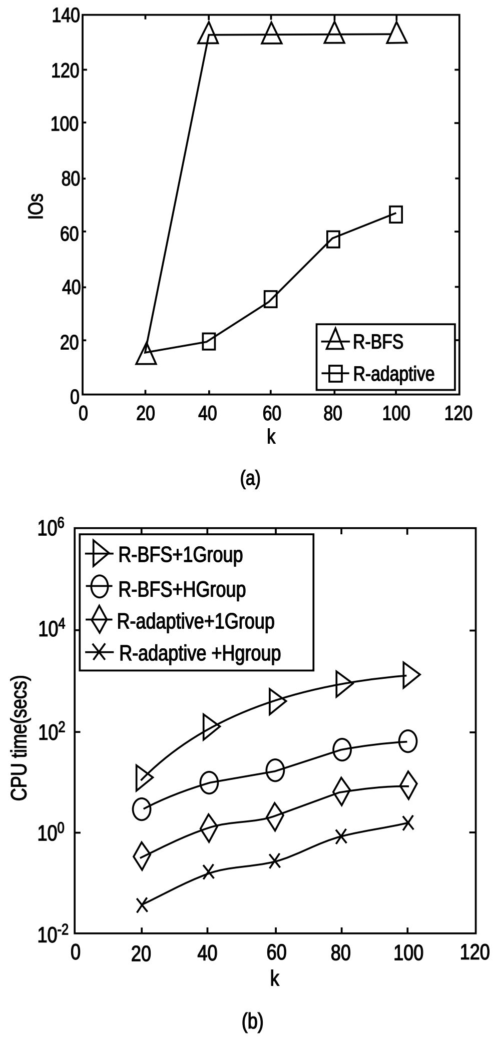
(b)



(c)

Fig. 7. Information loss with different approaches on (a) NA, (b) TG, and (c) Skew data sets.





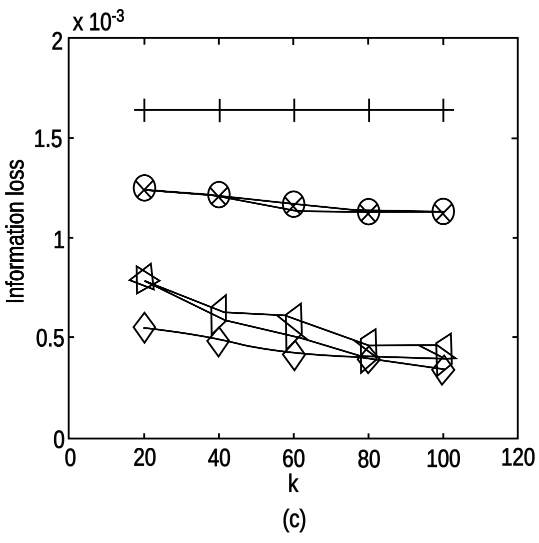


Fig. 8. Experimental results with varying k on the Skew data set. (a) I/O cost. (b) CPU time. (c) Information loss.

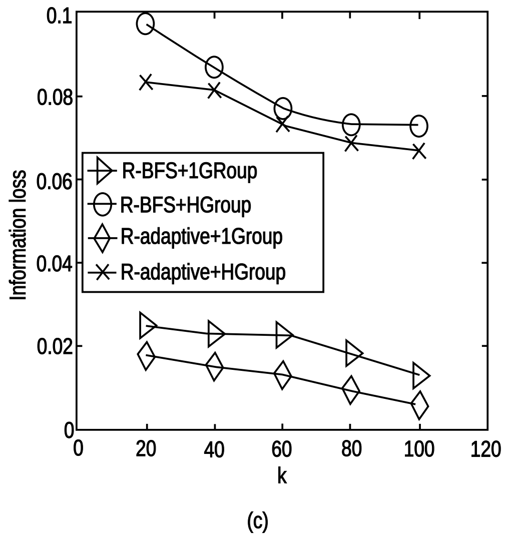
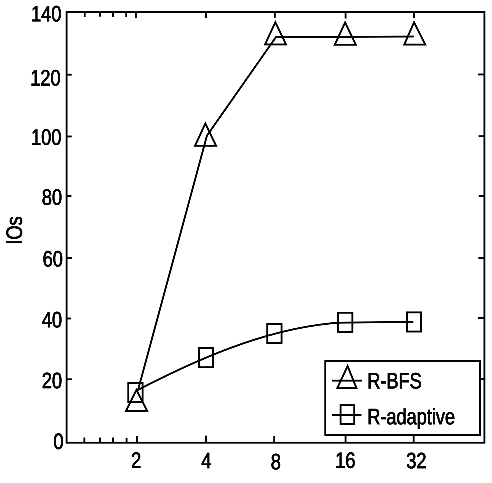
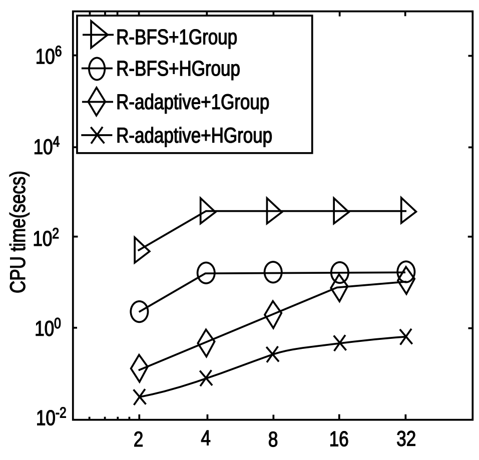


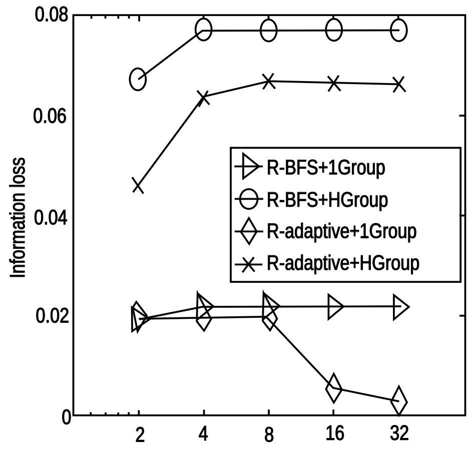
Fig. 9. Experimental results with varying k on the CA data set. (a) I/O cost. (b) CPU time. (c) Information *loss.*



(a)

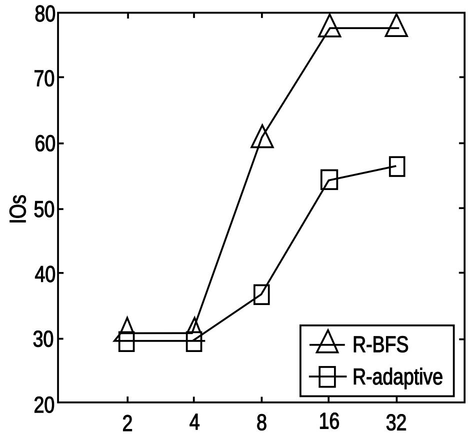


(b)

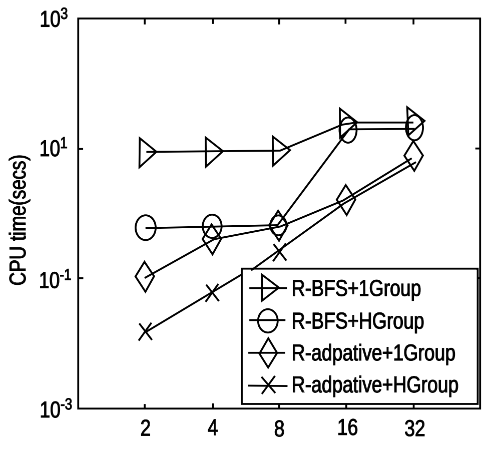


(c)

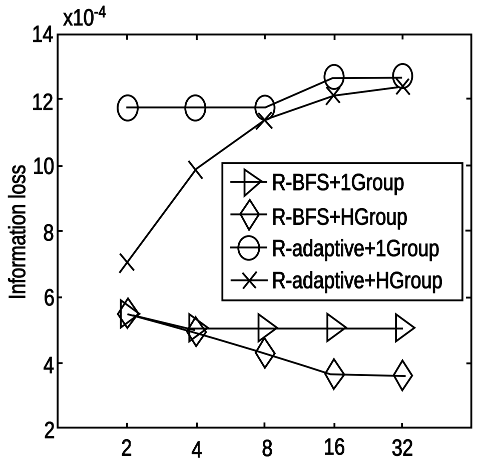
Fig. 10. Experimental results with varying on the CA data set. (a) I/O cost. (b) CPU time. (c) Information loss.



(a)

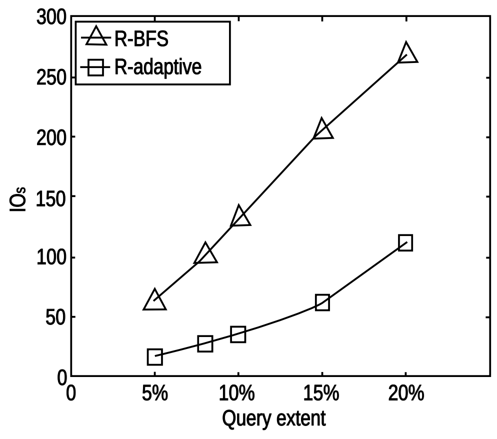


(b)

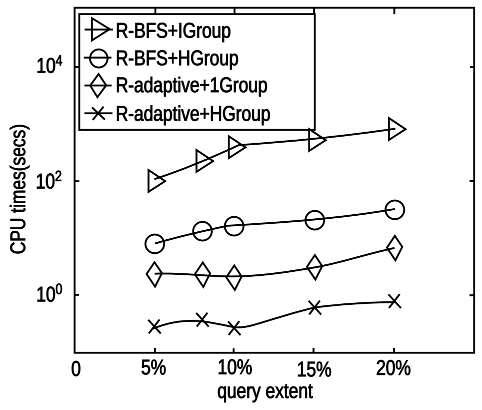


(c)

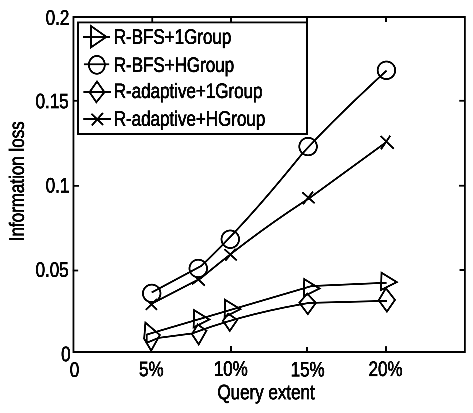
Fig. 11. Experimental results with varying on the Skew data set. (a) I/O cost. (b) CPU time (c) Information loss.



(a)

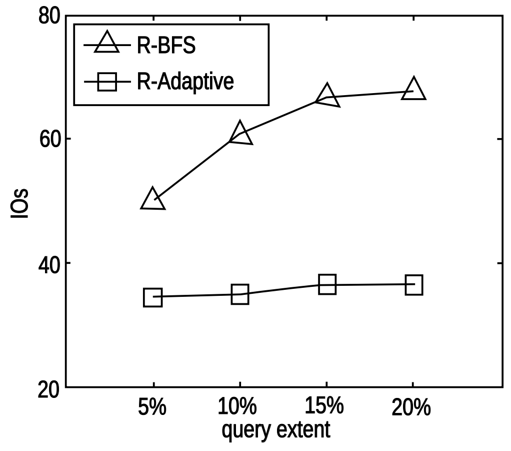


(b)

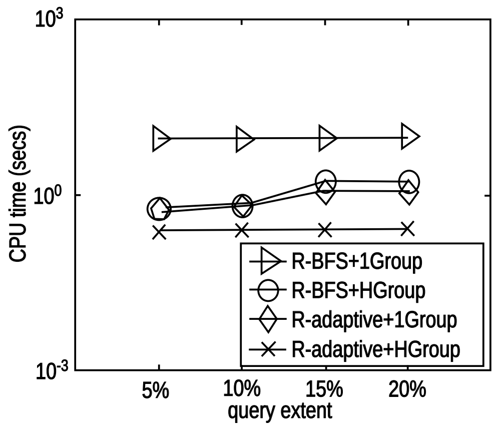


(c)

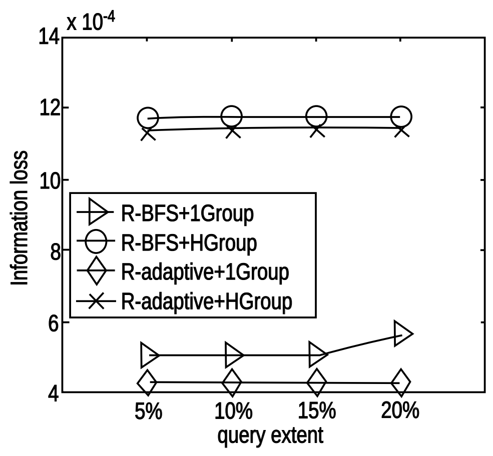
Fig. 12. Experimental results with varying query range on the CA data set. (a) I/O cost. (b) CPU time. (c) Information loss.



(a)

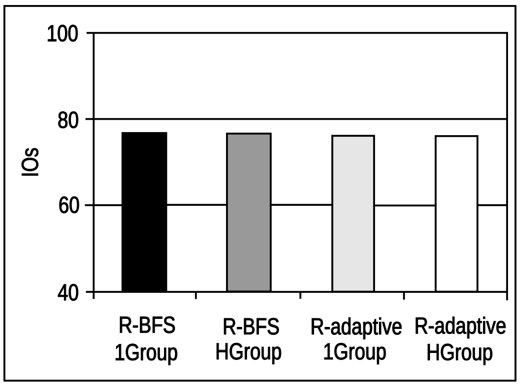


(b)

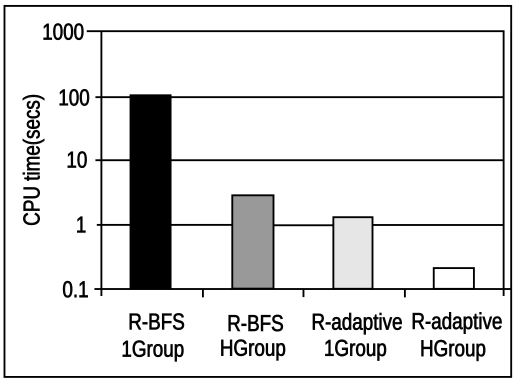


(c)

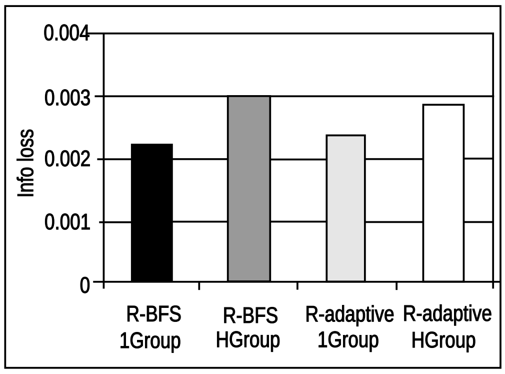
Fig. 13. Experimental results with varying query range on the Skew data set. (a) I/O cost. (b) CPU time. (c) Information loss.



(a)



(b)



(c)

Fig. 14. Experimental results on Skew data set with objects of rectangular shape. (a) I/O cost. (b) CPU time. (c) Information loss.

32. In Fig. 10, we plot the I/O cost, CPU time, and information loss for the four algorithms on the CA data set. The trends on the I/O cost and CPU time in this set of experiments are similar to those in the experiments with varying *k*.The experimental results are similar to that of CA. We do not show the similar results on NA and TG data sets here due to space limit.

G. *Experimental Results with Varying Query Size*

The results on I/O cost, CPU time, and information loss on the CA data set are shown in Fig. 12. First, similar to the previous two sets of experiments, the I/O cost of the algorithms increases as the query range gets larger.

In terms of CPU time, the trend is similar, i.e., all the algorithms take longer to run as the query gets larger. Therefore, in practice, if small response time is strictly required. We can see that the trend of our approaches on objects of rectangular shape is similar to that of point data sets.

VII. RELATED WORK

The motivation of this work is very similar to the recent work of Jermaine et al. The focus *t* 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. Density based clustering approaches, such as CLARANS and DBSCAN are another popular clustering definition. Our work is also related to the data summarization techniques such as histogram Specifically, Finally, sampling-based techniques could be applied to derive concise representations for the results of range queries. The NP-hardness result for two dimensions, the IGroup algorithm, the R-tree-based algorithms, the extensions, as well as the experiments, are all new in this paper.

VIII. CONCLUSION

A new concept, that of concise range queries, has been proposed in this paper, which simultaneously addresses the following thre First, it reduces the query result size significantly as required by the user. e problems of traditional range queries. Second, although the query size has been reduced, the usability of the query results has been actually improved. Finally, we have designed R-tree-based algorithms so that a concise range query can be processed much more efficiently than evaluating the query exactly, especially in terms of I/O cost. The concept of concise range queries is quite general.

One can imagine that it could naturally extend to deal with moving objects uncertainty and fuzziness in data etc. However, designing efficient and effective query processing algorithms for these types of data remains a challenging open problem.

REFERENCES

[1] X. Lin, Y. Yuan, Q. Zhang, and Y. Zhang, “Selecting Stars: The k Most Representative Skyline Operator,” Proc. Int’l Conf. Data Eng.(ICDE), 2007.

[2] C. Jermaine, S. Arumugam, A. Pol, and A. Dobra, “Scalable