International Journal on Future Revolution in Computer Science & Communication Engineering
Volume: 4 Issue: 11
ISSN: 2454-4248
01 – 06

Effective and Efficient Algorithms for Concise Range Queries

K. Soni Sharmila,
Assistant Professor,
Department of CSE,
D.N.R. College of Engineering and Technology.
sonisharmila.kadimi@gmail.com

A. S. N. V. Bhogesh,
Assistant Professor,
Department of CSE,
D. N. R. College of Engineering and Technology.
areti.bhogesh@gmail.com

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. 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

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 [1], 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, Jermaine et al. [2] argued to return concise representations of the final query results in every possible stage of a long-running query evaluation. However, the focus of [2] is on join queries in the relational database and the approximate representation is a random sample of the final query results. The goal of this work is different and random sampling is not a good solution for our problem.

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, it represent a point set using a collection of bounding boxes and their associated counts as a concise representation of the point set. It 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. Note that the definition above can be easily extended to a set of objects of arbitrary shapes, rather than just points. In particular, in Section 4, I will apply the same notion on a set of rectangles. But for now I will focus on point sets. Clearly, for fixed dimensions, the amount of bytes required to represent R is only determined by its size k.

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/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.

It 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. Unfortunately, I prove that finding

 ISSN: 2454-4248
01 – 06

the optimal representation with minimum information loss is an NP-hard problem. Therefore, I 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.

For 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. Moreover, it is equally the same for applications with PCs over the Internet. In these scenarios, the response time is a very critical factor for attracting users to choose the service of a product among different alternatives, e.g., Google Map versus MapQuest, since long response time may blemish the user experience. This is especially important when the query results have large scale.

Second, client's 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. 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. If I return a small representation of the whole query results, there is also the potential of reducing the processing cost on the server and getting around this lower bound.

II. Problem definition

The purpose of this project deals with the, 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. Too much information may do more harm than good, which is commonly known as the information overload problem. 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. Ideally, one would like to have a concise representation of minimum information loss.

III. System Implementation

Here focusesing on the problem of finding a concise representation for a point set P with minimum information loss. In one dimension, a simple dynamic programming algorithm finds the optimal solution in polynomial time [3]. However, this problem becomes NP-hard in two dimensions. Then, I settle for efficient heuristic algorithms for two or higher dimensions. The above BFS traversal treats all nodes alike in the R-tree and will always stop at a single level. But, intuitively, I should go deeper into regions that are more "interesting," i.e., regions deserving more user attention.

These regions should get more budgets from the k bounding boxes to be returned to the user. Therefore, I would like a quantitative approach to measuring how "interesting" a node in the R-tree is, and a corresponding traversal algorithm that visits the R-tree adaptively. In the algorithm R-Adaptive [4], I start from the root of the R-tree with an initial budget of k, and traverse the tree top-down recursively. Suppose I are at a node u with budget k, and u has b children u1, . . . ,ub those MBRs are either completely or partially inside Q. Let the counts associated with them be n1; . . . ; nb. Specifically, if BR(ui) is completely inside Q, I set ni ¼ nui ; if it is partially inside, I compute ni proportionally as in

$$n_u \cdot \frac{\text{Area}(\text{MBR}(u) \cap Q)}{\text{Area}(Q)}$$

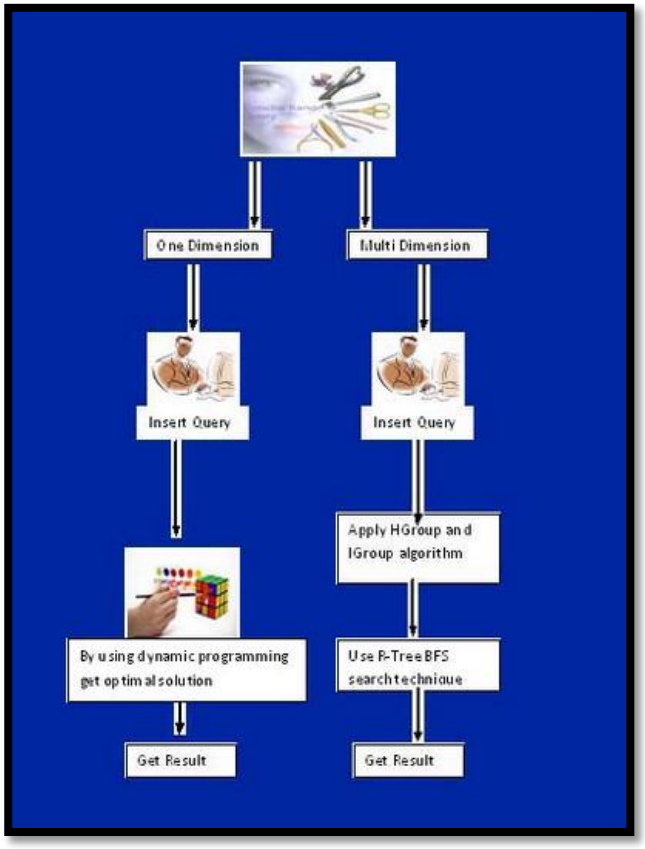

Fig: System Architecture

IV. Optimal Solution in One Dimension

 ISSN: 2454-4248
01 – 06

We first give a dynamic programming algorithm for computing the optimal concise representation for a set of points P lying on a line. Later, in Section 3.3, I will extend it to higher dimensions, leading to an efficient heuristic [7]. Let p1,,…,pn be the points of P in sorted order. Let $P_{i,j}$ represent the optimal partitioning underlying the best concise representation, i.e., with the minimum information loss, for the first i points of size $j, i \geq j$ optimal solution is simply the concise representation for $P_{n,k}$ and $P_{n,k}$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., I should always create a group with consecutive points without any point from another group in-between.

V. Hardness of the Problem in 2D

Not surprisingly, like many other clustering problems, for a point set P in IR2, the problem of finding a concise representation of size k with minimum information loss is NP-hard. Also, similar to other clustering problems in euclidean space, the NP-hardness proof is quite involved. It give a carefully designed construction that gives a reduction from PLANAR 3-SAT to the concise representation problem. In the classical 3-SAT problem, there are n Boolean variables x1; . . . ; xn and m clauses C1; . . . ; Cm, where each clause is a disjunction of three variables or their negations. The problem is to decide if there exists an assignment of variables such that all clauses evaluate to true. Assume that I map each variable and each clause to a vertex of an undirected graph G. Then, two nodes u and v in graph G are connected if and only if u represents a clause that contains the variable (or its negation) represented by v. If G is planar, the problem is known as PLANAR 3-SAT. It is an NP-complete problem [8]. The problem remains NP-complete even if each variable and its negation appear in at most three clauses. Note that in this case, any vertex in G has degree at most 3. Given an instance of PLANAR 3-SAT, I construct a point set P as follows: First, since any vertex in G has degree at most 3, I can find a planar embedding of G such that all the vertices lie on grid points and all the edges follow nonintersecting grid paths. Such an embedding can be found in O(n+m) time and the resulting embedding has area $O(n+m)^2$ [9].

VI. Heuristics for Two or More Dimensions

Given the hardness result, it is very unlikely that it can find an optimal concise representation R for a given point set P in polynomial time in two or more dimensions. Thus, in this section, I try to design efficient heuristic algorithms that produce an R with low information loss, although not minimum. Since the 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. However, since the object function makes a big difference in different clustering problems, the heuristics designed for other clustering problems do not work for our case. The k-anonymity problem does share the same object function with us, but the clustering constraint there is that each cluster has at least k points, while I require that the number of clusters is k. These subtle but crucial differences call for new heuristics to be tailored just for the concise representation problem.

Given the optimal algorithm in one dimension, a straightforward idea is to use a function $IR^d \rightarrow$IR to map the points of P from higher dimensions down to one dimension. Such an ordering function must somehow preserve the proximity relationships among the points. Many techniques exist for such a purpose (see [10] for a detailed review), among them the space-filling curves [11] have proved to be a popular choice. A space-filling curve traverses the space in a predetermined order. The most widely used space-filling curve is the Hilbert curve. The hth-order Hilbert curve has 2hd cells in d dimensions and visits all the cells in the manner shown in Fig. 5. Each cell will be assigned a Hilbert value in sequence starting from 0, and all points falling inside the cell will get the same Hilbert value. Our Hilbert-curve based algorithm, called HGroup, is shown in Algorithm 1. The basic idea is to first compute the Hilbert value for each point in P, and sort all points by this value, mapping them to one dimension

VII. Algorithm 1: The algorithm HGroup

Compute the Hilbert value h(p_i) for each point $p_i \epsilon$ P Sort P by h(p_i) and map it to one dimension; Find the partitioning P using dynamic programming; Build the concise representation R for P and return;

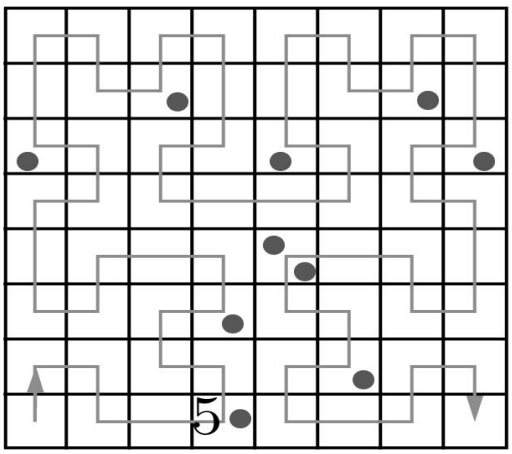

Fig: The third-order Hilbert curve in two dimensions

 ISSN: 2454-4248
01 – 06

VIII. Algorithm 2: Algorithm IGroup

It presents another, more direct algorithm in two or more dimensions. It is an iterative algorithm that finds the k groups P1; . . . ; Pk, one at a time. This algorithm is called as I Group. In each iteration, it start with a seed, randomly chosen from the remaining points, and greedily add points into the group one by one. In the ith iteration, I first initialize the group Pi to include only the seed. Let U be the set of remaining points. When I stop adding points and obtain a Pi, I record that together with its $\tilde{L}(p_i)$ achieved in the end, and call it a candidate. To improve quality, I carry out the same process with a number of different seeds, and choose the best candidate that has attained the lowest $\tilde{L}(p_i)$. Then, I proceed to the next group P_{i+1} . Finally, for the last group Pk, I simply put all the remaining points into it. I give the details of the complete algorithm IGroup in Algorithm 2.

ALGORITHM 2: I-Group

U← P;

S←number of seeds to try;

For i=1,….,k-1 do

$\acute{L}_{best}$ =∞;

U' ←U;

For j=1,…..,s do

U← U'

p_s ←randomly chosen seed from U;

$P_i' \leftarrow p_s$;

While true do

Let p=argmin_p $\acute{L}(P_i' \cup \{p\})$;

If $\acute{L}(P_i' \cup \{p\}) < \acute{L}(P_i')$ then

$P_i' \leftarrow P_i' \cup \{p\}$;

U←U-{p};

Else break;

If $\acute{L}(P_i') < \acute{L}_{best}$ then

$\acute{L}_{best} \leftarrow \acute{L}(P_i')$;

$p_i \leftarrow P_i'$;

U←U-p_i;

Output p_i;

There is k - 1 iterations in the algorithm. In each iteration, it check each of the n points and choose the best one to add to the current group. In the worst case, I 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 I-Group is $O(skn)^2$.

IX. Query processing with r-trees

In order to use the algorithms 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. After obtaining the complete query results P ⋂ Q, the server then partitions P ⋂ Q into k groups and returns the concise representation. However, as the main motivation to obtain a concise answer is exactly because P ⋂ Q is too large, finding the entire P ⋂ Q and running the algorithms. It presents algorithms that process the concise range query without computing P ⋂ Q in its entirety. The idea is to first find k^1 bounding boxes, for some k^1 > k, that collectively contain all the points in P ⋂ Q by using the existing spatial index structure on P. Each of these bounding boxes is also associated with the count of points inside. Then, I run a weighted version of the algorithm and grouping these k^1 bounding boxes into k larger bounding boxes to form the concise representation R. Typically $k^1 \ll |P \cap Q|$, so it could expect significant savings in terms of I/O and CPU costs as compared with answering the query exactly. Therefore, adopting concise range queries instead of the traditional exact range queries not only solves the bandwidth and usability problems, but also leads to substantial efficiency improvements.

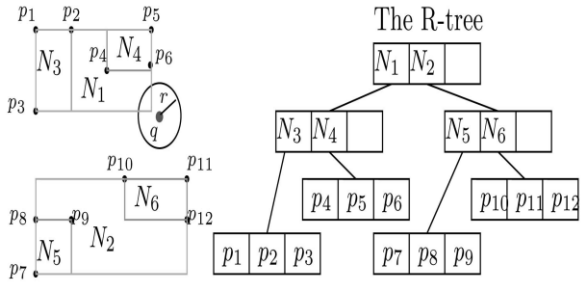

Fig: R-Tree

The algorithms presented in this section in general work with any space partitioning index structure; for concreteness, I will proceed with the R-tree, which is arguably the most widely used spatial index structure. The R-tree [13] and its variants (R-tree in particular [14]) all have the following structure. Suppose the disk block size is B. We first group_ B points in proximity area into a minimum bounding rectangle (MBR); these points will be stored at a leaf on the R-tree. These MBRs are then further grouped together level by level until there is only one left. Each node u in the R-tree is associated with the MBR enclosing all the points stored below, denoted by MBR(u). Each internal node also stores the MBRs of all its children. An example of the R-tree is illustrated in Fig. 6. Different R-

 ISSN: 2454-4248
01 – 06

tree variants only differ in the rule show the MBRs or points are grouped together. The standard range query Q can be processed using an R-tree as follows: I start from the root of the R-tree, and check the MBR of each of its children. Then, I recursively visit any node u whose MBR intersects or falls inside Q. When I reach a leaf, I simply return all the points stored there that are inside Q. In this section, I in addition assume that each node u in the R-tree also keeps nu, the number of the points stored below its sub tree. Such counts can be easily computed and maintained in the R-tree.

X. Algorithm R-BFS

The straightforward way to find k0 such MBRs is to visit the part of the R-tree inside Q in a BFS manner, until I reach a level where there are at least _k MBRs. I call this algorithm R-BFS. In particular, for any node u whose MBR is complete inside Q, I directly return MBR(u) together with nu. For any node u whose MBR is partially inside Q, I return the intersection of the MBR(u) and Q, while the associated count is estimated as

$$n_i.\frac{\text{Area (MBR } (u_i) \cap Q)}{Area\ (Q)}$$

assuming uniform distribution of the points in MBR(u).

XI. Algorithm R-Adaptive

The BFS traversal treats all nodes alike in the R-tree and will always stop at a single level. But, intuitively, it should go deeper into regions that are more "interesting," i.e., regions deserving more user attention. These regions should get more budget from the k bounding boxes to be returned to the user. Therefore, I would like a quantitative approach to measuring how "interesting" a node in the R-tree is, and a corresponding traversal algorithm that visits the R-tree adaptively.

In the algorithm R-Adaptive, I start from the root of the R-tree with an initial budget of k=αk, and traverse the tree top-down recursively. Suppose I am at a node u with budget k, and u has b children u1; . . . ; ub whose MBRs are either completely or partially inside Q. Let the counts associated with them be n1; . . . ; nb. Specifically, if MBR(ui) is completely inside Q, I set $n_i = n_u$; if it is partially inside, I compute ni proportionally as in (4).

ALGORITHM 3: Recursive call visit(u,k)
Let $u_1,\ldots\ldots,u_b$ be u's chidren whose MBRs are inside or partially inside Q; Let n_i =number of points inside MBR(u_i)$\cap Q$; If b>=k then Output MBR(u_i) $\cap Q$ with n_i for all I; Return; Let A_i =Area(MBR(u_i) $\cap Q$);

Compute k_i as in for all i=1,……,b;
For i=1,…..,b do
If k_i =1 then
Output MBR(u_i) $\cap Q$ with n_i
Else
Visit(u_i, k_i);

XII. Performance Analysis

Input data

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/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. Collected spatial data are provided as an input.

Performance is measured in terms of the output provided by the application. Requirement specification plays an important part in the analysis of a system. Only when the requirement specifications are properly given, it is possible to design a system, which will fit into required environment. It rests largely with the users of the existing system to give the requirement specifications because they are the people who finally use the system. This is because the requirements have to be known during the initial stages so that the system can be designed according to those requirements. It is very difficult to change the system once it has been designed and on the other hand designing a system, which does not cater to the requirements of the user, is of no use.

XIII. Result Analysis

This paper implemented two base algorithms, HGroup and IGroup, as well as the two R-tree traversal algorithms RBFS and R-Adaptive. Specifically, I used the R*-tree [14] to index all the points in the data set. The IGroup algorithms first traverse the R-tree to produce a number of MBRs and feed them into the base algorithms, so I have, in total, two combinations: R-BFS + IGroup, R-BFS + HGroup, R-Adaptive + IGroup, and R-Adaptive + HGroup. The straw-men I compare against are the k-means clustering algorithm and the MinSkew histogram [15]. In particular, for k-means and MinSkew, I first obtain all the data points in the query region via R-tree, and then conduct either of the two methods to obtain clusters/partitions.

 ISSN: 2454-4248
01 – 06

It first did some test queries to see if the concise representation indeed gives the user some intuitive high level ideas about the query results. The resulting concise representations can be overlapping with each other. However, this is usually not a problem for the user perception, as long as I render the higher density rectangles in front of the lower density ones. the average information loss per point in the concise representations returned, where I observe the following. First, the two IGroup-based variants produce much better results than the two HGroup-based variants. The explanation is possibly that, since the Hilbert curve imposes a linear order on the 2D data set, some important geometric properties are lost. While Hilbert-curve-based clustering algorithms [12] generally work well, our problem is fundamentally different from traditional clustering problems. Thus, simply extending the 1D dynamic programming algorithm to two dimensions by Hilbert curves does not work as well as IGroup, which is directly designed for two dimensions. Second, coupled with the same grouping algorithm, R-Adaptive works much better than R-BFS. This means that visiting the R-tree more selectively does not only save the traversal cost, but also leads to better quality of the results. In addition, I observe that larger ks improve the quality for all the algorithms. This is intuitive, since as the size of the concise representation increases, it becomes closer and closer to the exact results.

XIV. Conclusion

A new concept that of concise range queries has been proposed in this paper, which simultaneously addresses the following three problems of traditional range queries. First, it reduces the query result size significantly as required by the user. The reduced size saves communication bandwidth and also the client's memory and computational resources, which are of highest importance for mobile devices. Second, although the query size has been reduced, the usability of the query results has been actually improved. The concise representation of the results often gives the user more intuitive ideas and enables interactive exploration of the spatial database. Finally, I 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. This concept, together with its associated techniques presented here, could greatly enhance user experience of spatial databases, especially on mobile devices, by summarizing "the world in a nutshell."

XV. Future Work

The basic idea of representing range queries in a concise format .However, the only technical contents in are the basic 1D dynamic programming algorithm and the HGroup algorithm. 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.

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 Approximate Query Processing with the dbo Engine," Proc. ACM SIGMOD, 2007.

[3] G. Ghinita, P. Karras, P. Kalnis, and N. Mamoulis, "Fast Data Anonymization with Low Information Loss," Proc. Int'l Conf. Very Large Data Bases (VLDB), 2007.

[4] G. Aggarwal, T. Feder, K. Kenthapadi, S. Khuller, R. Panigrahy, D. Thomas, and A. Zhu, "Achieving Anonymity via Clustering," Proc. Symp. Principles of Database Systems (PODS), 2006.

[5] J. Xu, W. Wang, J. Pei, X. Wang, B. Shi, and A.W.-C. Fu, "Utility- Based Anonymization Using Local Recoding," Proc. ACM SIGKDD, 2006.

[6] C. Bo¨hm, C. Faloutsos, J.-Y. Pan, and C. Plant, "RIC: Parameter- Free Noise-Robust Clustering," ACM Trans. Knowledge Discovery from Data, vol. 1, no. 3, pp. 10-1-10-28, 2007.

[7] R.T. Ng and J. Han, "Efficient and Effective Clustering Methods for Spatial Data Mining," Proc. Int'l Conf. Very Large Data Bases (VLDB), 1994.

[8] D. Lichtenstein, "Planar Formulae and Their Uses," SIAM J. Computing, vol. 11, no. 2, pp. 329-343, 1982.

[9] R. Tamassia and I.G. Tollis, "Planar Grid Embedding in Linear Time," IEEE Trans. Circuits and Systems, vol. 36, no. 9, pp. 1230-1234, Sept. 1989.

[10] H.V. Jagadish, B.C. Ooi, K.-L. Tan, C. Yu, and R. Zhang, "iDistance: An Adaptive B+-Tree Based Indexing Method for Nearest Neighbor Search," ACM Trans. Database Systems, vol. 30,no. 2, pp. 364-397, 2005.

[11] H. Samet, The Design and Analysis of Spatial Data Structures. Addison-Wesley Longman Publishing Co., Inc., 1990.

[12] B. Moon, H.v. Jagadish, C. Faloutsos, and J.H. Saltz, "Analysis of the Clustering Properties of the Hilbert Space-Filling Curve," IEEE Trans. Knowledge and Data Eng., vol. 13, no. 1, pp. 124-141, Jan. 2001.

[13] A. Guttman, "R-Trees: A Dynamic Index Structure for Spatial Searching," Proc. ACM SIGMOD, 1984.

[14] N. Beckmann, H.P. Kriegel, R. Schneider, and B. Seeger, "The R-Tree: An Efficient and Robust Access Method for Points and Rectangles," Proc. ACM SIGMOD, 1990.

[15] T. Zhang, R. Ramakrishnan, and M. Livny, "BIRCH: An Efficient Data Clustering Method for Very Large Databases," Proc. ACM SIGMOD, 1996.