**ANDROID SHOPPING LOCATION IDENTIFIER**

**Abstract:**

Given a set of points P and a query set Q, a *group enclosing query* (GEQ) fetches the point p\_ 2 P such that the maximum distance of p\_ to all points in Q is minimized. This problem is equivalent to the Min-Max case (minimizing the maximum distance) of aggregate nearest neighbor queries for spatial databases. This work first designs a new exact solution by exploring new geometric insights, such as the minimum enclosing ball, the convex hull and the furthest voronoi diagram of the query group. To further reduce the query cost, especially when the dimensionality increases, we turn to approximation algorithms. Our main approximation algorithm has a worst case p2-approximation ratio if one can find the exact nearest neighbor of a point. In practice, its approximation ratio never exceeds 1.05 for a large number of data sets up to six dimensions. We also discuss how to extend it to higher dimensions (up to 74 in our experiment) and show that it still maintains a very good approximation quality (still close to 1) and low query cost. In fixed dimensions, we extend the p2-approximation algorithm to get a (1 + ǫ)-approximate solution for the GEQ problem. Both approximation algorithms have O (log N +M) query cost in any fixed dimension, where N and M are the sizes of the data set P and query group Q. Extensive experiments on both synthetic and real data sets, up to 10 million points and 74 dimensions, confirm the efficiency, effectiveness and scalability of the proposed algorithms, especially their significant improvement over the state-of-the-art method.

**Existing System:**

While being general enough to cover different aggregate operators, its generality also means that important opportunities could be overlooked to optimize query algorithms for specific operators. For instance, the state of the art is limited by heuristics that may yield very high query cost in certain cases, especially for data sets and queries in higher (more than two) dimensions. Motivated by this observation, this work focuses on one specific aggregate operator, namely the MAX, for the aggregate nearest neighbor queries in large databases and designs methods that significantly reduce the query cost compared to the MBM (Minimum Bounding Method) algorithm from.

Following the previous instance when studying a specific aggregate type for aggregate nearest neighbor queries (e.g., group nearest neighbor queries for the SUM operator, we designate a query name, the group enclosing query (GEQ), for an aggregate nearest neighbor query with the MAX operator. An example of the GEQ problem is illustrated.

More applications could be listed to demonstrate the usefulness of this query and more examples are available from the prior study. Intuitively, in contrast to many existing variants of the nearest neighbor queries that ask for the best answer of the average case the GEQ problem searches for the best answer of the worst case. This problem becomes even more difficult if the query group is large as well.

**Proposed System:**

This work presents new, efficient algorithms, including both exact and approximate versions, for the GEQ problem that significantly outperform the state of the art, the MBM algorithm. Specifically, • We present a new exact search method for the GEQ problem in Section 4 that instantiates several new geometric insights, such as the minimum enclosing ball, the convex hull and the furthest voronoi diagram of the query group, to achieve higher pruning power than the MBM approach.

• We design a √2-approximation (worst case approximation ratio in *any dimensions*) algorithm in Section 5.1, if one can find the exact nearest neighbor of a point and the minimum enclosing ball of Q. Its asymptotic query cost is O (log N + M) in *any fixed* *dimensions*. Our idea is to reduce the GEQ problem to the classical nearest neighbor search by utilizing the center of the minimum enclosing ball for Q.

• We extend the above idea to a (1+ǫ)-approximation algorithm in any fixed dimension in Section 5.2. This algorithm has a strong theoretical interest and it also achieves the optimal O (log N+M) query cost in any fixed dimension.

• We extend the same idea from the √2-approximate algorithm to much higher dimensions in Section 5.3, since it is impossible to find the exact nearest neighbor *efficiently* and the exact minimum enclosing ball in high dimensions in practice.

• We discuss the challenges when Q becomes large and disk-based in Section 6.1, and show how to adapt our algorithms to handle this case efficiently. We also present an interesting variation of the GEQ problem, the constrained GEQ.

• We demonstrate the efficiency, effectiveness and scalability of our algorithms with extensive experiments. These results show that both our exact and approximate methods have significantly outperformed the MBM method up to 6 dimensions. Beyond 6 dimensions and up to very high dimensions (d = 74), our approximate algorithm is still efficient and effective, with an average approximation ratio that is close to 1 and very low IO cost.

**Implementation Modules:**

1. Aggregate Nearest Neighbor
2. Approximate Nearest Neighbor
3. Min Max Nearest Neighbor
4. Nearest Neighbor

**Aggregate Nearest Neighbor:**

The state of the art method for the GEQ problem is the Minimum Bounding Method (MBM) from. The principal methodology adopted by the MBM is the triangle inequality. It is a heuristic method that has O (N + M) query cost. In practice, its query cost has only been studied in the two-dimensional space. Our experiments over a large number of data sets in Section 7 suggests that the MBM algorithm may lead to high query cost for large data sets and more importantly, its performance degrades significantly with the increase of the dimensionality.

**Approximate Nearest Neighbor:**

In fact, it is easy to see that any exact search method for the GEQ problem will inevitably suffer from the curse of the dimensionality, since the classical nearest neighbor search is a special instance of the GEQ problem (when Q has only one point). Hence, for data sets in high dimensions, similar to the motivation of doing approximate nearest neighbor search instead of retrieving the exact nearest neighbor in high dimensions (where almost all exact methods degrade to the expensive linear scan of the entire data set), finding efficient and effective Approximation algorithms are the best alternative.

**Min Max nearest Neighbor:**

R-tree and its variants have been the most widely deployed indexing structure for the spatial database, or data in multi-dimensions in general. Intuitively, R-tree is an extension of the B-tree to higher dimensions. Points are grouped into minimum bounding rectangles (MBRs) which are recursively grouped into MBRs in higher levels of the tree. The grouping is based on data locality and bounded by the page size. An example of the R-tree is illustrated in Figure 2. Two important query types that we leverage on R-tree are nearest neighbor (NN) queries and range queries.

NN search has been extensively studied, and many related works therein. In particular R-tree demonstrates efficient algorithms using either the depth-first or the best-first approach. The main idea behind these algorithms is to utilize branch and bound pruning techniques based on the relative distances between a query point q to a given MBR N (e.g., minmaxDist, minDist).

**Nearest Neighbor:**

Unfortunately, the worst case query costs are not logarithmic when R-tree is used for NN or range search (even for approximate versions of these queries). To design theoretically sound algorithms with logarithmic costs for our problem, we need a space partition tree with the following properties : (1) The tree has O(N) nodes, O(log N) depth and each node of the tree is associated with at least one data point. (2) The cells have bounded aspect ratio. (3) The distance of a point to a cell of the tree can be computed in O (1). Aria et.al designed a modification of the standard kd-tree called the Balanced Box Decomposition (BBD) tree which satisfies all these Properties and hence can answer (1 + ǫ)-approximate nearest neighbor queries in O((1/ ǫd) log N) and (1 + ǫ)- approximate range search queries in O((1/ǫd) + log N). BBD-tree takes O (N log N) time to build. We use BBD trees in the design of the optimal (1 + ǫ)-approximation algorithm with the logarithmic query complexity for Solving the GEQ problem. For nearest neighbor search in high dimensions, all exact methods will eventually degrade to the expensive linear scan of the entire data set and one has to adopt efficient and effective approximate algorithms.

The BBD-tree also becomes impractical for large data sets in high dimensions. In this case, we could use the iDistance index for exact nearest neighbor retrieval (in still relatively low dimensions), or Medrank and LSH-based methods (locality sensitive hashing) (e.g., the latest development represented by the LSB-tree) for the approximate versions in very high dimensions. Since our idea in designing the approximate algorithms for solving the GEQ problem is to reduce it to the basic nearest neighbor search problem, our approach could leverage on all these techniques for the nearest neighbor search and benefit by *any advancement* in this topic. This is a very appealing feature of our approximation algorithm and makes it extremely flexible and simple for adaptation.

**SYSTEM REQUIREMENTS:**

**HARDWARE REQUIREMENTS:**

* System : Pentium IV 2.4 GHz.
* Hard Disk : 40 GB.
* Floppy Drive : 1.44 Mb.
* Monitor : 15 VGA Colour.
* Mouse : Logitech.
* Ram : 512 Mb.

**SOFTWARE REQUIREMENTS:**

* Operating system : Windows XP.
* Coding Language : Java 1.6
* Tool Kit : Android 2.2
* IDE : Eclipse