RESEARCH ARTICLE OPEN ACCESS

# NKS Search in Multidimensional Dataset with and without using User Preference

Shimna P.T [1], Dilna V.C [2]
AWH Engineering College, Calicut university
Department of Computer science & Engineering
Kuttikkatoor, Kozhikode, India

**ABSTRACT**
Conventional spatial queries, such as range search and nearest neighbor retrieval, involve only conditions on objects' geometric properties. Today, many modern applications call for novel forms of queries that aim to find objects satisfying both a spatial predicate, and a predicate on their associated texts. For example, instead of considering all the restaurants, a nearest neighbor query would instead ask for the restaurant that is the closest among those whose menus contain "steak, spaghetti, brandy" all at the same time. we develop a new access method Preference Tree with user preference. In Preference Tree time and distance is using as parametrs.We develop another method for without user preferences by Z curve method. The proposed techniques outperform in query response time significantly.

***Keywords :—*** Nearest Neighbor Search, Keyword Search, Spatial Index.

## I. INTRODUCTION

A spatial database manages multidimensional objects (such as points, rectangles, etc.), and provides fast access to those objects based on different selection criteria. The importance of spatial databases is reflected by the convenience of modeling entities of reality in a geometric manner. For example, locations of restaurants, hotels, hospitals and so on are often represented as points in a map, while larger extents such as parks, lakes, and landscapes often as a combination of rectangles. Many functionalities of a spatial database are useful in various ways in specific contexts. For instance, in a geography information system, range search can be deployed to find all restaurants in a certain area, while nearest neighbor retrieval can discover the restaurant closest to a given address.

In Fig. 1, suppose there are a set of businesses whose locations (represented by squares) and service lists (a set of keywords) are registered in the online yellow pages of a local search service provider. When a GPS-enabled smartphone user wants to find a nearby restaurant to have a piece of pizza and a cup of coffee, she may send the local search server two keywords, coffee and pizza. Based on the user's current location (e.g., the point q in Fig. 1) derived from the smartphone and the two query keywords, business p1 is returned by the server. Note that although businesses p3 and p4 are closer to q than p1, they do not satisfy the keyword constraint.

In many real applications, the query workload may vary from time to time, and the system may encounter a burst of queries (e.g., queries invoked by a particular event). In this scenario, the system throughout is poor if a large number of queries are processed one by one. Motivated by this, a large body of existing work have been devoted to investigate how to improve the system throughout with the batch query processing techniques such that a large number of queries in the queue can be processed with a reasonable delay.

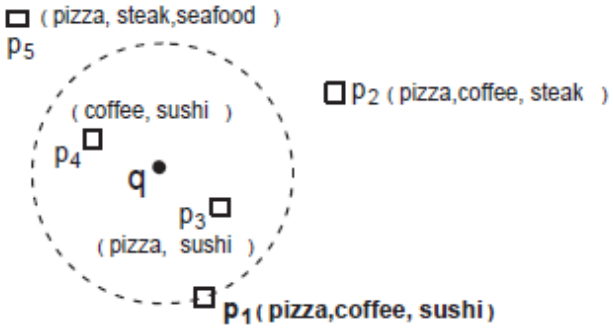


**Figure 1:** Online Yellow Pages Example

In this paper, we propose Preference Tree to enable fast processing for NKS queries. In particular, we develop an exact user preference and without user preference that always retrieves the optimal top-k results. Preference Tree uses a set of hashtables and inverted indexes to perform a localized search. The hashing technique is inspired by Locality Sensitive Hashing, which is a state-of-the-art method for nearest neighbor search in high-dimensional spaces. Unlike LSH-based methods that allow only approximate search with probabilistic guarantees, the index structure in Preference Tree supports accurate search. A single round of search in a hashtable yields subsets of points that contain query results, and Z curve method used to explores each subset using a fast pruning-based algorithm without user preference. We evaluate the performance of Preference Tree and Z curve method on both real and synthetic datasets. The empirical results reveal that Preference Tree will efficiently work than Z curve method in the basis of user preference otherwise considering time time as parametrer Z curve method will work more faster than preference tree.

## II. RELATED WORKS

With the ever-increasing popularity of services such as Google Earth and Yahoo Maps, as well as other geographic applications, queries in spatial databases have become increasingly important in recent years. novel type of query called the mclosest keywords (mCK) query: given m keywords provided by the user, the mCK query aims at finding the closest tuples (in space) that match these keywords. While such a query has various applications, our main interest lies in that of a search by document. The bR*-tree[2] is an extension of the R*-tree. Besides the node MBR, each node is augmented with additional information. A straightforward extension is to summarize the keywords in the node. With this information, it becomes easy to decide whether m query keywords can be found in this node.

There are many applications which are quality sensitive and need to efficiently and accurately support near neighbor queries for all query ranges. propose a novel indexing and querying scheme called Spatial Intersection and Metric Pruning (SIMP)[5]. It efficiently supports r-near neighbor queries in very high dimensional spaces for all query ranges with 100% quality guarantee and with practical storage costs.

Efficient method to answer top-k spatial keyword queries. To do so, we introduce an indexing structure called IR2-Tree (Information Retrieval R-Tree)[3] which combines an R-Tree with superimposed text signatures. present algorithms that construct and maintain an IR2-Tree, and use it to answer top-k spatial keyword queries. method to efficiently answer top-k spatial keyword queries, which is based on the tight integration of data structures and algorithms used in spatial database search and Information Retrieval (IR). In particular, our method consists of building an Information Retrieval R-Tree (IR2Tree), which is a structure based on the R-Tree. At query time an incremental algorithm is employed that uses the IR2-Tree to efficiently produce the top results of the query.

A hybrid search strategy between LSH-based search and linear search for r-NN in high dimensional space. By integrating an auxiliary data structure into LSH hash tables, we can efficiently estimate the computational cost of LSH-based search for a given query regardless of the data distribution. This means that we are able to choose the appropriate search strategy between LSH-based search and linear search to achieve better performance. Moreover, the integrated data structure is time efficient and fits well with many recent state-of-the-art LSHbased approaches[4].

fundamental application of locating geographical resources and propose an efficient tagcentric query processing strategy. In particular, we aim to find a set of nearest co-located objects which together match the query tags. Given the fact that there could be large number of data objects and tags, efficient search algorithm that can scale up in terms of the number of objects and tags. Further, to ensure that the results are relevant[7].

Keyword-based search in text-rich multi-dimensional datasets facilitates many novel applications and tools. In this paper, we consider objects that are tagged with keywords and are embedded in a vector space. For these datasets, queries that ask for the tightest groups of points satisfying a given set of keywords.Novel method called ProMiSH (Projection and Multi Scale Hashing) [1]that uses random projection and hash-based index structures, and achieves high scalability and speedup.

## III. PREFERENCE TREE

In this section hash-based index structures using distance and time based pruning and achieves high scalability and speedup. We present an exact and an approximate version of the algorithm and architecture.

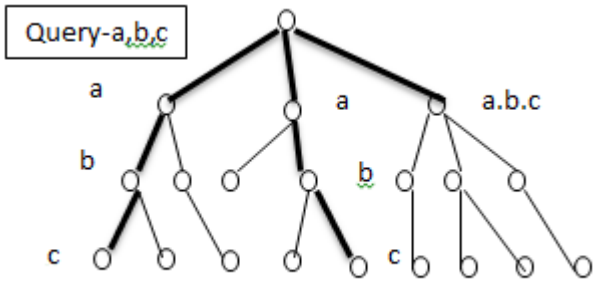


Figure 2 Preference Tree

### *A. Architecure*

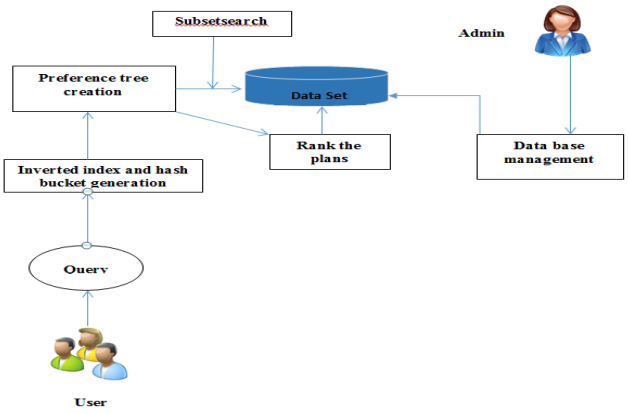


Figure 3 Architecture of Preference Tree

**Preference tree creation:** nodes are created in the basis of user preference and also checking the distance is less than threshold.

**Rank the plan:** in the basis of sum of distance and sum of time which path getting shortest distance and time.

### *B.* Algorithm

**Algorithm**
**In:** location ,point of interest ,place name
**Index creation**
Get all points containing the features

First Colum keyword
Second colum point id
**Hash table generation**
Create hash vector for each point which selected in previous step
1.User to select the origin
Either users current location or user can pick the point from google map.
2. Find all preference point from index
3.Get all points with i th preference
4. If (distance <threshold)
5.Get all node of levels
6.Generate hash vector using preference
7.Checking the node with hash vector
Untilfull fill the hash vector
8.Rank the plan
9.stop

Algorithm 2 Preference Tree using user preference

## IV. Z CURVE WITHOUT USER PREFERNCE

In this section inverted index is compressing using Z curve to get more space reduction and also producing new algorithm for without user preference. Furthermore, as the Z curve is based on the conventional technology of inverted index, it is readily incorporable in a commercial search engine that applies massive parallelism, implying its immediate industrial merits

### *A. Compression scheme*

Compression is already widely used to reduce the size of an inverted index in the conventional context where each inverted list contains only ids. In that case, an effective approach is to record the gaps between consecutive ids, as opposed to the precise ids. For example, given a set S of integers {2,3,6,8}, the gap-keeping approach will store {2,1,3,2} instead, where the i-th value ($i \geq 2$) is the difference between the i-th and (i−1)-th values in the original S. As the original S can be precisely reconstructed, no information is lost. The only overhead is that decompression incurs extra computation cost, but such cost is negligible compared to the overhead of I/Os. Note that gap-keeping will be much less beneficial if the integers of S are not in a sorted order. This is because the space saving comes from the hope that gaps would be much smaller (than the original values) and hence could be represented with fewer bits. This would not be true had S not been sorted.

Compressing an SI-index is less straightforward. The difference here is that each element of a list, a.k.a. a point p, is a triplet (idp,xp,yp), including both the id and coordinates of p. As gap-keeping requires a sorted order, it can be applied on only one attribute of the triplet. For example, if we decide to sort the list by ids, gap-keeping on ids may lead to good space saving, but its application on the x- and y-coordinates would not have much effect. To attack this problem, let us first leave out the ids and focus on the coordinates. Even though each point has 2 coordinates, we can convert them into only one so that gap-keeping can be applied effectively. The tool needed is a space filling curve (SFC) such as Hilbert- or Z-curve.

SFC converts a multidimensional point to a 1D value such that if two points are close in the original space, their 1D values also tend to be similar. As dimensionality has been brought to 1, gap-keeping works nicely after sorting the (converted) 1D values. For example, based on the Z-curve2, the resulting values, called Z-values, of the points in Figure 1a are demonstrated in Figure 5 in ascending order. With gap keeping, we will store these 8 points as the sequence 12,3,8,1,7,9,2,7. Note that as the Z-values of all points can be accurately restored, the exact coordinates can be restored as well. Let us put the ids back into consideration. Now that we have successfully dealt with the two coordinates with a 2D SFC, it would be natural to think about using a 3D SFC to cope with ids too. As far as space reduction is concerned, this 3D approach may not a bad solution.

The problem is that it will destroy the locality of the points in their original space. Specifically, the converted values would no longer preserve the spatial proximity of the points, because ids in general have nothing to do with coordinates. If one thinks about the purposes of having an id, it will be clear that it essentially provides a token for us to retrieve (typically, from a hash table) the details of an object, e.g., the text description and/or other attribute values. Furthermore, in answering a query.

### *B. Architecure*

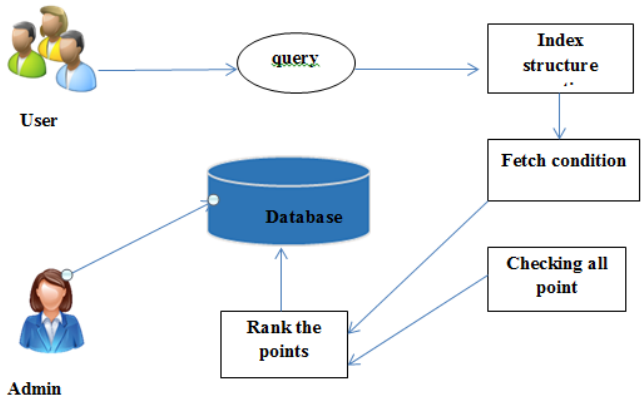


Figure 4. Architecture of Z curve method

**Fetch condition:** set a radious r and compare with all points in Z curve using point of intrest.add the point in list if the point satify condition.condition is point will present within the radious in Z curve and satisfy all point of inttrest.

**Rank the points:**after generating hash vector until full fill the pont of intrerest.

### *C. Algorithm*

**Algorithm**
**Input:** location ,point of interest
1. User select the origin

2. Convert point into Z curve
3. Set radious r
4. Get all points from Z curve satisfying radious
5. Fetch point()
   All point satisfy the condition radious as well as all point of intrestAdd to list
6. Generate hash vector
   Until hash vector satisfy until full fill the point of interest
7. Generate plan
8. Otherwise
   Radious = r+Δr
9. Repeat step 4
10. Stop

Algorithm 2. Z curve without user preference

## V. CONCLUSION AND FUTURE WORK

In this paper, we proposed solutions to the problem of topk nearest keyword set search in multi-dimensional datasets. We proposed a novel index called Preference Tree. hash-based index structures using distance and time based pruning and achieves high scalability and speedup. We present an exact and an approximate version of the algorithm. we have remedied the situation by developing an access method called the Z curve without user preference. Not only that the Z curve is fairly space economical, but also it has the ability to perform keyword-augmented nearest neighbor search in time that is at the order of dozens of milliseconds.

In the future, we plan to explore other scoring schemes for ranking the result sets Then, each group of points can be scored based on distance between points and weights of keywords. Furthermore,the criteria of a result containing all the keywords can be relaxed to generate results having only a subset of the query keywords.using satellite imading to find the nearest locations.

## REFERENCES

[1] Vishwakarma Singh, Bo Zong, Ambuj K. Singh, Nearest Keyword Set Search in Multi-dimensional Datasets, *IEEE Transactions on Knowledge and Data Engineering* in 2016.

[2] D. Zhang, Y. M. Chee, A. Mondal, A. K. H. Tung, and M. Kitsuregawa, "Keyword search in spatial databases: Towards searching by document," *in ICDE*, 2009, pp. 688–699

[3] I. De Felipe, V. Hristidis, and N. Rishe, "Keyword search on spatial databases," *in ICDE*, 2008, pp. 656–665

[4] Ninh Pham, IT University of Copenhagen Denmark ndap@itu.d, Hybrid LSH: Faster Near Neighbors Reporting in High-dimensional Space, *in ICDE*, 2010.

[5] Vishwakarma Singh, Santa Barbar, Ambuj,K. Singh, SIMP: Accurate and Efficient Near Neighbor Search in High Dimensional Spaces, *EDBT 2012*, March 26–30, 2012.

[6] Yufei Tao, Cheng Sheng, Fast Nearest Neighbor Search with Keywords, *IEEE Transactions on Knowledge and Data Engineering* , in 2012.

[7] Dongxiang Zhang, Beng Chin Ooi, Anthony K. H. Tung, Locating Mapped Resources in Web 2.0 , SIGMOD, 2013.