International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878 (Online), Volume-8 Issue-2, July 2019

CONNEKT: Co-Located Nearest Neighbor Search using KNN Querying with K-D Tree

S. Sharmiladevi, S. Siva Sathya, Naveen Kumar

Abstract*: Data about entities or objects associated with geographical or location information could be called as spatial data. Spatial data helps in identifying and positioning anyone or anything globally anywhere across the world. Instances of various spatial features that are closely found together are called as spatial co-located patterns. So far, the spatial co-located patterns have been used only for knowledge discovery process but it would serve a wide variety of applications if analyzed intensively. One such application is to use co-location pattern mining for a context aware based search. Hence the main aim of this work is to extend the K-Nearest Neighbor (KNN) querying to co-located instances for context aware based querying or location-based services (LBS). For the above-said purpose, co-located nearest neighbor search algorithm namely "CONNEKT" is proposed. The co-located instances are mapped onto a K-dimensional tree (K-d tree) inorder to make the querying process efficient. The algorithm is analyzed using a hypothetical data set generated through QGIS.

Index Terms*: Co-located pattern mining, K-Nearest neighbor, K-dimensional tree, Location-based services.

I. INTRODUCTION

Spatial data has been widely used by Geographical information system (GIS) and other geo-location or spatial services [1][2][3]. Points, lines, polygons and other geographic and geometric data primitives are some of the data types of spatial data. These spatial data can also be used for mapping location, storing an object as metadata or to locate user's devices by making use of communication systems. Spatial data is often classified as scalar or vector data. Unambiguous information with respect to geographical or spatial location can be obtained using either of these data. In general, spatial data corresponds to the orientation, location, size and form of an entity present on Earth. Spatial data can also have properties that cater to additional knowledge about the entity being considered. Geographic Information Systems (GIS) or other Geo specific software applications can be used to acquire, envision, exploit and examine geospatial data [4][5]. The importance of spatial data mining is growing with the occurrence and significance of voluminous spatial dataset repositories of remote-sensing images [6], context aware mobile applications [7], satellite imagery [8], medical data and crime data [9], three dimensional maps, traffic data [10] and many more. A spatial co-location pattern provides association between spatial features that are usually located together [11][12][13]. These patterns also provide useful insights for location-based services and GIS. The co-location pattern recognition process discovers the subsets of feature often located together in a cluster of Boolean spatial features. For example, the examination of an ecology dataset may disclose the association between symbiotic species. Unlike association rule problem, there is no natural notion of transactions in spatial data sets since they are embedded in continuous geographic space. Spatial co-location patterns are used for various applications, but this work aims at using co-location patterns for a completely different perspective. Till now the spatial co-located patterns have been used only for making better inferences, but it has a wide range of usage. Using co-located pattern set for a context aware based search or LBS is one such usage. Location based services (LBS) considers the geographical location of the device. LBS have been usually used for either information or entertainment purpose. The main aim of LBS service provider is to discover the location of user because LBS rely on user's location. Context awareness and location awareness are property of mobile devices [14]. When location focuses mainly on how certain processes around a contributing device operate, context can be applied more conveniently on mobile users, especially with the users of smart phones. A context aware search is one in which the user does a search with respect to his location. For example, a user may want to find a theatre, a mall and a restaurant which are located together, within 10 km from his location. These kinds of spatial queries require the location of the individual instances and finally perform a join operation to know whether such pattern is found within the user's range. The above-mentioned process is complex; hence a lookup in the already mined co-located patterns can be done to find out if such an instance is present within the range. The purpose of this work is to use the mined co-located patterns for a KNN [15] querying. The K nearest neighbor (KNN) is used to find the nearest K neighbors for a query from a given dataset. The co-located patterns are got by using a co-location pattern mining technique. The mined patterns can be represented in a K-d tree, which increases the efficiency of querying. A K- d tree is a space-partitioned data structure for arranging points in a K-dimensional space and is very useful for various applications, such as searches involving a multidimensional search key (e.g. range searches and nearest neighbor searches).

Revised Manuscript Received on 30 July 2019.
***** Correspondence Author
S. Sharmiladevi*, Department of Computer Science, Pondicherry University, Puducherry, India.
S. Siva Sathya, Department of Computer Science, Pondicherry University, Puducherry, India.
Naveen Kumar, Department of Computer Science and Engineering, KL University, Guntur , India.

K-d trees are efficient in bulk-loading situations. The result obtained from this querying process is the set of co-located instances with respect to users' location.

The rest of the paper is organized as follows: Section-2 deals with the related work, especially focusing on the state of the art techniques related to Location based services, KNN queries, K-d trees, Co-location Mining, etc., Section-3 projects the proposed work namely CONNEKT, Section-4 deals with the experimental setup and analysis, Section-5 presents the results and finally Section-6 concludes the paper.

II. LITERATURE SURVEY

This section gives a detailed review of the various techniques that are required to perform the co-located nearest neighbor search. The basics of location-based services, KNN query for Location Based Search, the data structure used namely the K-d tree, and the Co-location Mining details are dealt in detail.

A. Location based services

Location Based Services (LBS) [16] are mobile phone oriented services and these services take into consideration the device's geographical location. LBS typically provide information or entertainment. The service providers are interested in finding frequently co-located services together in the proximity. With the knowledge of co-located patterns, they do queries. A master system which provides guidance for tourists is proposed in [17][18]. This system makes use of a multi-agent web service system along with an ontological database system, to provide travel-oriented information. A location-based service supported Information system is developed here, i.e. a multi-agent master system acts as a travel information supplier and path deviser for tourist by making use of an image recognizing methodology along with Google maps. This application has received an overall satisfaction score of 80%. The designed system allows user to have fun during travelling by making use of a QR/Barcode reader, cloud database, a smartphone with integrated GPS function, to discover the necessary web services. For testing purpose New Taipei City in Taiwan was used. The two most popular trends in the emergence of electronic commerce are LBS in mobile communication and the personalization of information recommendation. But still, only one of these two trends have been focused by various prior researches. In [19], the application of LBS is integrated with recommendation technologies to give a location-based service recommendation model (LBSRM) and develop an experimental system to generate and measure the reliability of LBSRM. In the recommendation model, an adaptive method including long-term and short-term preference adjustment is conducted because of the accumulation and variation of preference, and also to strengthen the result of recommendation. A mobile platform framework has been proposed for location based service (LBS) and geospatial information (GI) in [20]. The framework is an integration of a number of state-of-the-art techniques with geospatial data resources that provides robust and highly available LBS.

B. K-d Tree

K-d trees [21] are useful data structures for several applications, such as searches involving a multidimensional search key (e.g. range searches and nearest neighbor searches). K-d trees are a special case of binary space partitioning tree. The following are some of the few methods available for constructing K- d tree. The spatial join queries are streamlined by exploiting the framework processing assets as per the qualities of spatial data in [22]. A spatial join query undertaking is isolated into a few sub-spatial join inquiries, which can keep running in parallel to enhance the query execution. A state-of-the-art method for computing approximate nearest neighbor (ANN) fields is the Propagation-assisted K-d tree [23]. In this method, each query patch needs descending search in the K-d tree and propagation search in the nearby patches. In this work the observations reveal that the query patches in the edge region need descending search, while other query patches only need propagation search. Therefore, this method requires less execution time. The edge propagation K-d tree was proposed to quickly compute ANNs. The experiments carried out on public data set shows that the search method is 2 to 3 times faster than the propagation-assisted K-d tree search method at nearly the same accuracy. A simple and flexible multidimensional data structure has been introduced in [24].The quad K-d tree is a general purpose and hierarchical data structure for the storage of multidimensional points. Quad K- d trees include both the point quad trees and K-d trees as particular cases and therefore it constitutes a general framework for the study of fundamental properties of trees similar to them. Besides, quad K-d trees can also be tuned by means of insertion heuristics and bucketing techniques to obtain trade-offs between their costs in time and space. It has been shown that the performance of random quad K-d trees is between random quad trees and random K-d trees with respect to IPL (internal path length) and number of empty sub-trees.

C. K Nearest Neighbor Search

KNN search [25] is used for finding the nearest neighbors with respect to the given location. Several modifications of KNN search has been introduced in literature. The following are some of the techniques available for performing this search. A novel quick KNN look up strategy utilizing an orthogonal search tree has been introduced in [26]. The proposed procedure makes use of an orthogonal scan tree for data collection utilizing an orthonormal premise assessed from the data set. To discover the KNN for a user given query point from the dataset, the estimations of the query point are mapped onto the orthogonal vectors and a node elimination methodology is applied for pruning unwanted nodes. For a node, which can't be removed, a point elimination inequality is additionally used to dismiss impossible data points. Experimental outcomes demonstrate that this strategy has suitable overall performance on finding KNN for query factors and always requires less calculation time than current KNN seek algorithms, mainly for a dataset with a huge quantity of data points or deviation. The previous KNN research made assumptions that the query handling calculations have access to their spatial index, and also that the spatial databases have to be queried are local;

e.g., R-trees. When KNN queries are focused on the spatial databases which are remote and work independently then this suspicion will be invalid. As it is conceivable to duplicate a few or the entire spatial entities which are present in remote databases or in a neighbourhood database and manufacture a different file structure for them, such an option is impossible when the size of database is big, or if there are substantial number of spatial databases to be queried. A KNN query processing algorithm that retrieves the nearest neighbors for a given query point using one or more window queries is proposed in [25]. In order to determine the span to be exercised by the window queries two different methods are proposed. A multi-staged framework (MSF) for high performance and adaptable execution of massive concurrent continuous query processing is introduced in [27], which exploits pipeline strategy and improves the parallelism with GPU computing technology. And a GPU based shortest path network expending algorithm (GPU-SPNE) is presented, which uses threaded workload parallelism to improve the throughout and performance for massive continuous KNN computing. The k nearest neighbor (KNN) search is a critical task in various fields, for example, LBS and database area. Recently few techniques have been proposed to fasten KNN search for static nodes with GPUs. To assess large simultaneous queries towards mobile objects in spatial systems, a multi-staged framework MSF has been introduced to enhance the parallelism with multi-threaded innovation, which leaves the question handling into three concurrent stages for ceaseless KNN inquiry. Further, in-memory spatial network adjacent matrix, shortest path matrix and hash table structures are introduced to describe the road network topology and store the mobile objects. Under MSF system a GPU-SPNE algorithm is proposed to diminish the figuring expense of KNN queries by utilizing threaded workload parallelism. Test assessment demonstrates that GPU-SPNE calculation accomplishes a performance improvement around one to two requests of magnitude over its CPU partners, and still performs superior to the brute-force algorithm on GPU in all conditions Under MSF framework a GPU-SPNE algorithm is proposed to decrease the computing cost of KNN queries by using threaded workload parallelism. The design of parallel K-d tree construction and NNS on GPU for low and high dimensional spaces is proposed in [15]. The proposed algorithms are of comparable quality as traditional sequential counterparts on CPU, while achieving high speedups. Parallel algorithms for the construction of K- d tree and nearest neighbor search (NNS) on massively parallel architecture (MPA) of graphics processing unit (GPU) are proposed. Unlike previous parallel algorithms for K-d tree, for the first time, this parallel algorithm integrates high dimensional K-d tree construction and NNS on an MPA platform. The proposed massively parallel algorithms are of comparable quality as traditional sequential counterparts on CPU, while achieving high speed performance on both low and high dimensional K-d tree. Low dimensional K-d tree construction and NNS algorithms, presented in this paper, outperform their serial CPU counterparts. For high dimensional K-d tree, the speedup factors are even higher.

A partially specified nearest neighbor query search is used in [28]. The algorithm is based upon the k-d tree algorithms for partial match searches and nearest neighbor searches. The tree is assumed to be a non-homogeneous tree, in which the records are stored at the leaves.

D. Co-location pattern mining

This section presents some of the widely used Co-location pattern mining approaches.

Mining Co-located patterns enables to discover a group of spatial entities that are often found together. Spatial data presents a challenge for finding co-location patterns as the traditional data is considered discrete whereas the spatial objects are embedded in a continuous space. The existing work on this concept makes use of a distance threshold which is inefficient for spatial data sets with differing magnitudes of neighborhood distances, particularly for regional co-location pattern mining. Hence a hierarchical co-location pattern mining method that works well on both varied neighborhood distances and spatial heterogeneity is proposed in [29]. A K-nearest neighbor graph (KNNG) is chosen as an alternative for distance threshold, a “distance variation coefficient” is used as a new procedure to manage the mining operations and discover an individual neighborhood relationship graph for each region. The experimental outcome on both synthetic and real-world data sets demonstrates that this method is far more efficient to find these regional co-location patterns. A join-less approach is proposed in previous work, but as the data size increases, a large amount of computation time is spent to find co-location rules as the existing approaches are purely sequential. Hence a parallelized join-less approach which finds the spatial neighbor relationship inorder to identify co-location instances and co-location rules is proposed in [16]. It finds the spatial neighbor relationships using a grid-based approach. The computation cost of finding the co-location patterns is decreased by using map-reduce framework. As traditional data mining methods are generally inefficient for spatial data mining, various other spatial data mining methods are specifically developed. a model for finding the most commonly occurring co-location patterns in spatial datasets which utilizes clustering technique before mining the data for rules is proposed in [25]. This algorithm namely, SigCPM [30] overcomes the shortcomings of the existing grid-based approaches.

III. PROPOSED SYSTEM

The main aim of this research is to propose and model an efficient search technique for spatial queries involving more than one feature using a Kd-tree to represent the mined co-located patterns and adopts a KNN algorithm to do a lookup in the tree. Hence a complete framework namely CONNEKT has been proposed that involves the use of multiple algorithms and tools for realizing an efficient multi-feature search tool. The architectural design, implementation and results are presented in this section.

A. Co-Located Nearest Neighbor Search using KNN Querying with K-d Tree (CONNEKT)

The existing location based services provide nearest neighbor for a single feature. Hence, CONNEKT aims at providing nearest neighbor with more than one feature.

Since the join operations in spatial database are costlier, the co-located instance obtained from co-location pattern mining can be used for this purpose. The spatial co-located patterns can be used for a KNN query involving more than one feature. For instance if the user goes for an outing with his/her family, and each of the family member wants to visit different places, like the grandmother wants to visit a temple, the children want to go to a park, wife wants to go to a movie and the whole family wants to dine in a restaurant at the end of the day. Then the user can find a location or a place which has all these features located together. The user might also specify the range within which he wants to find a place having all these features. For these types of spatial queries we can do a lookup in the mined co-located patterns which has been mapped into a K- d tree. So, a K nearest neighbor search is performed on the K- d tree that is constructed for that pattern. This pattern involves n features that are mentioned by the user while querying. The following are the definitions for certain keywords used throughout the proposed work:

- Feature: A feature represents various public entities present in a city. For example, hotel, temple, parks etc.
- Co-ordinates: It represents the GPS location of a feature. In this work (X, Y) is used as co-ordinate value.
- Pattern: It is a combination of one or more features.eg. [hotel, temple] is a pattern of size 2, [temple, park, hotel] is a pattern of size 3.
- Instance: One or more occurrence of a pattern or feature. For example, a feature hotel can have many instances like A hotel, B hotel etc..., and a pattern [hotel, park] can have many instances like [A hotel, B park], [X hotel, I park] etc…

B. CONNEKT Methodology

The proposed model aims at finding the co-located neighbors for a given user location. A complete flow diagram of CONNEKT is given in Fig.1 and the algorithms given in Table. I-IV. In order to accomplish this, the co-ordinates of features are needed. These values are loaded to the co-location pattern mining algorithm [25] and the result produced by this algorithm is a set of co-located patterns, where each pattern has many instances. The co-ordinates of features and the co-located instances are stored in two databases namely “Feature” and “Pattern” respectively. By making use of these databases the instance calculation procedure is done, which is a necessary step required to map the co-located instances into a k-d tree. A nearest neighbor search is carried out on the k-d tree by using the inputs given by the user; the resultant of this search is a set of co-located neighbors.

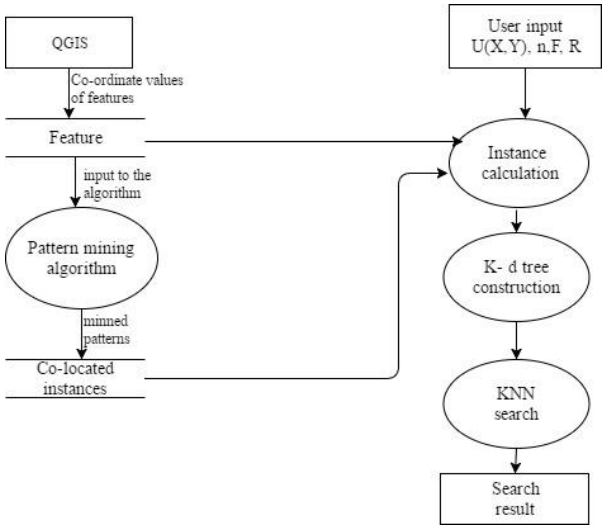

Fig. 1: Flow Diagram of CONNEKT

The algorithm used for KNN Querying of Co-located Instances using K-d Tree requires the following parameters as input.

- **User location U (X, Y):** The users’ current coordinate value or any coordinate value for which the user wants to find the nearest neighbor set involving one or more features that are co-located.
- **No. of features selected N:** This value lets us know whether the search is going to involve one or two or three features. This value depicts the pattern size.
- **Features F:** The feature that is being involved in the search is mentioned here. It can be one feature or two features or three features. For example, Hotel, park, bus stand etc…
- **Radius R: T**his value is used to find the search area. The circumference within which the search has to be done.
- **Database containing features:** This table contains the values of the randomly generated features.
- **Database containing mined co-located instances:** The table containing the feature is loaded to a co-location pattern mining algorithm [25] and the result got from this is a set of co-located instances.

The architecture of CONNEKT comprises of four important modules which are described below:

i. **User input module**:

The user has to provide the following input via the user module for the co-located nearest neighbor search.: u (x, y), N, F,R . These inputs along with the database containing feature and mined co-located patterns is used for discovering the nearest neighbors .Initially the value N is checked ; if this value is ≥ 1 then an instance calculation procedure follows before constructing the K-d tree and performing the search.

ii. **Instance calculation:**

The instance calculation module given in Table.II is an essential procedure for the co-located nearest neighbor search. The mined co-located patterns got as output from the co-location pattern mining algorithm will be in the form of doublets, triplets, etc. based upon the pattern size. Each pattern will have the feature type and its instances will have their corresponding id e.g.,

[Hotel, park] = [34, 10] [39, 16], etc. Here [hotel, park] is a pattern of size two and [34, 10] is called as an instance of the pattern, 34 is the id of a hotel located in some (X_1, Y_1) location and 10 is the id of a park located in some neighborhood (X_2, Y_2) location. We calculate a (X, Y) value for each instance of the patterns by finding the average of x and y values of the features using their corresponding id, $X = {X_1 + X_2}/{2}, Y = {Y_1 + Y_2}/{2}$.

This (X, Y) values calculated for the instance of the pattern is used for constructing a K-d tree for that particular pattern.

iii. **K-d tree construction module:**

For the construction of a K-d tree in algorithm given in Table.III we need the co-ordinate value for each instance of a pattern that was calculated in the instance calculation step, and a depth value (odd or even) to identify the splitting axis. Tree leaves store points or co-ordinate values (one or several points in each leaf). Each point is stored in one and only one leaf, each leaf stores at least one point. The set of all instances of a given pattern (pattern size may vary) forms a list hereinafter referred to as "pat-inst-list". For a particular query involving 2 or more features, such a list is generated with the instance id and their respective coordinate values. The coordinate values of the list are sorted and a ftsplit divide both the space and the pat-inst-list into two discrete parts. A successive split is made from the root node to one of the leaves, and eliminates part of the list (and space) until only a small part of the list (and space) is left. The list of instances is split alternatively either by x-coordinate or by y-coordinate in the following manner:

- Split by x-coordinate: If depth value is even, then split pointlist using the median M's x co-ordinate value intoP1 (left of or on M) and P2 (right of M)

- Split by y-coordinate: If depth value is odd, then split pointlist using the median M's y co-ordinate value into P1 (below or on M) and P2 (above M). This splitting repeats iteratively until all the points are assigned to a leaf node.

iv. **Nearest neighbor module:**

Nearest neighbor search as given in Table.IV is performed on the k-d tree that was constructed in the tree construction step. The root N of the k-d tree along with the user's point and radius is given as input. A two-dimensional array called neighbor is initialized to null in the beginning. Then the node N is checked to see wether N.left and N.right are NULL, if it is true then the point is N.point, which is stored in the neighbor array. If the left and right child are not null then check to see if the u(axis).value ≤ n.value (i.e. the user location value of the corresponding splitting axis is almost less than or equal to the root node N.value). When this condition holds true, a search is performed on the left sub-tree by making use of the radius value to determine the range of the search. The algorithms for the above said functions are given in the next subsection.

C. CONNEKT Algorithm

Table. I: CONNEKT procedure

Procedure CONNEKT(U=(X,Y) , n, F,R) // outputs the nearest neighbors to users location
Begin
If (n≥1)
Call Search ()
Else
Print "Entry not valid"
Endif
Search ()
If n=1
K-d tree ()
KNN ()

Else if(n≥2)
Instance calc ()
K-d tree ()
KNN ()
Endif
End

Table II: Instance Calc procedure

Instance calc (n, F)*// To compute the average position of the co-located pattern instance*

$f_1, f_2, \ldots\ldots\ldots\ldots, f_i \in F$

Begin

//summation of the x co-ordinate values involved in the patterns instance

$f_i X = {\sum_{k=0}^{n} x_i}/{n}$

//summation of the y co-ordinate values involved in the patterns instance

$f_i Y = {\sum_{k=0}^{n} y_i}/{n}$

End

Table III: K-d Tree Construction Procedure

K-d tree (Pat-inst-list, depth) //*Tree construction*

Begin

//Axis will be selected based on depth value.

Depth=0

Pointlist= $I_{(x_1,y_1)}, I_{(x_2,y_2)}, I_{(x_3,y_3)}, \ldots\ldots\ldots\ldots I_{(x_n,y_n)}$

If Pointlist contains a single point

then return a leaf containing this point

else if depth is even *//split across x axis if depth is even*

sort the Pointlist and find the median M

then Split Pointlist via the median x-coordinate into P1 (left of or on M) and P2 (right of M)

else Split Pointlist via the median y-coordinate into P1 (below or on M) and P2 (above M)

v. left ← K-d tree (P1, depth+1)

v. right ← K-d tree (P2, depth+1)

Create a node v storing M,

v.leftchild = v. left *//set v. left as the left child of v*

v.rightchild = v. right *// set v. right as the right child of v*

return v

End

Table IV: KNN Query procedure

KNN (U:user point, N: node(root), R: radius) *//Nearest neighbor search using K-d tree*

Begin

Neighbor [][]=Null; *// A 2d array which stores the nearest neighbors*

If n. left =n. right=null

Then{leaf}

Neighbor [][]=n. point

Else

//if the users value corresponding to the splitting axis is less than the root node's value then search left subtree else search right subtree

If u(axis).value ≤ n.value then

Dosearch =left;

Else

Dosearch =right;

Endif

If(search==left)

If U(axis).value - r ≤ n.value then KNN (U, n. left, r)

If U(axis).value + r >n.value then KNN (U, n. right, r)

Else

If U(axis).value + r >n.value then KNN (U, n. right, r)

If U(axis).value - r ≤ n.value then KNN (U, n. left, r)

Endif

End

IV. EXPERIMENTAL SETUP AND ANALYSIS

A. QGIS-Datasets Generation

The experiments are performed on synthetic dataset. The dataset is generated using Random Point Generation algorithm in QGIS tool. This dataset consists of 10 features and they are hotel, park, temple, ATM, theater, departmental store, shopping mall, police station, bus stand, railway station. Each feature had 10 instances and a total of 100 instances are generated.

B. Co-location pattern mining

The synthetic dataset is given as input to a co-location mining algorithm called SigCPM [30].

The algorithm comprises of three major phases:

1. Transaction generation using clustering over datasets
2. Co-location pattern mining using interest measure calculation
3. Co-location rule generation using co-location patterns.

The output of this algorithm consists of possible co-located patterns along with the instances of each pattern. The values given as input to the co-location pattern mining algorithm is stored in a Fig.2 consisting of three columns. The column which contains the instance id is made as the primary key. One column for the feature type (1-hotel, 2-park, 3-temple, 4-ATM, 5-theater, 6-departmental store, 7-shopping mall, 8-police station, 9-bus stand, 10-railway station.), and the last column for storing the co-ordinates. An integer data-type is used for storing the id and feature. Point data-type is used for storing the co-ordinates.The mined co-located patterns are stored in a table consisting of two columns. The column which contains the pattern is of type integer and is made as the primary key. The column which consists of the instances of the corresponding pattern is of type integer array.

	key [PK] numeric	feature name	x point
1	1	1	(15.7474626767653,7.81287395749931)
2	2	1	(74.9483963465556,20.6166644766602)
3	3	1	(30.8266888640121,-11.3912762260906)
4	4	1	(30.8266888640121,-11.3912762260906)
5	5	1	(24.3441693620648,-6.72674185562474)
6	6	1	(51.828235169017,-17.6286124681337)
7	7	1	(47.5067334282519,20.7994946340604)
8	8	1	(51.5048763764144,-9.11868989230143)
9	9	1	(36.288741730212,-11.3562219557457)
10	10	1	(48.7504097691608,-0.492726348030384)
11	11	2	(43.0009423732877,6.5447220342031)
12	12	2	(29.6485050103896,-5.33895315027706)
13	13	2	(29.5713947268931,7.47969440570729)
14	14	2	(6.02338765663369,7.67186830376585)
15	15	2	(34.9012772063325,1.37944302244464)
16	16	2	(4.11076163289747,-2.75921473577985)
17	17	2	(17.3866051520345,20.6526040046251)
18	18	2	(20.0617715382447,0.64294531046129)
19	19	2	(7.52212614187333,18.9565457428012)
20	20	2	(35.1555989423348,5.20312421118009)
21	21	3	(68.3941616059777,15.3942034882731)

atch pad

Fig. 2: Input to Co-located pattern mining algorithm

C. User Interface Design

A user interface is created using JSP , for this purpose a server is used , which is apache tomcat. An open-source Java Servlet Container created by the Apache Software Foundation (ASF) is called as Tomcat Server. The database and the server are connected using the jdbc connection string "DriverManager.getConnection("jdbc:postgresql://localhost: 5432/features", "username", "password");" where 5432 is the postgreslocalhost port number, features is the database name which is followed by the username and password for the database. The Fig.3 shows the user interface that has been created for the user to give the input and Fig.4 shows the output window.

D. Performance Analysis

The algorithm has been analyzed for its time complexity in terms of big-O notation. Since CONNEKT is the first of its kind in terms of nearest neighbor search with multiple co-located features, it could not be compared with other such algorithms. The proposed technique is found to have a time complexity of $\Theta(\log N)$ which is very less compared to the time required by existing join based approach. In general, If the search is done by using join operators, then the time complexity for 2 features will be $\Theta(N^3)$ and $\Theta(N^4)$ for 3 features.

V. RESULTS

A. Search involving 2 features

The user can select any two features from the 10 features that are available. The range for the search is to be mentioned and finally the co-ordinate values with respect to which the search is carried out. Fig.3 represents the GUI for a search involving 2 features. The user has selected two features i.e. hotel and park. The co-ordinate entered is (15.6, -1.2). The range chosen is 20km. The output for this query is present in Fig.4. The result is a set of points where a hotel and park are found.

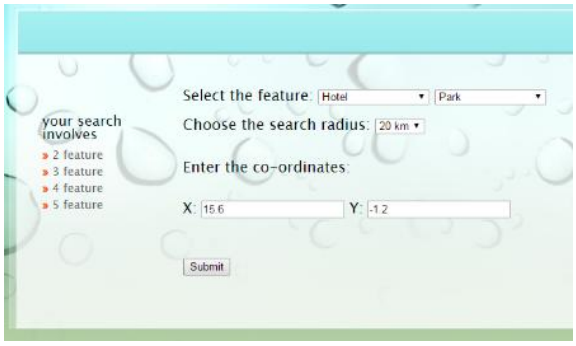

Fig.3: Input screen for 2 features

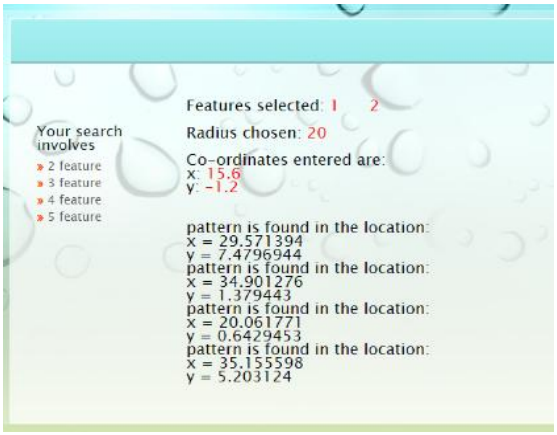

Fig.4: Output screen for the search involving 2 features

B. *Search involving 3 features*

This search is similar to that of the previous search involving 2 features. Here we have 3 features involved in the search (i.e. ATM, park, and shopping mall). The radius given is 7km. The co-ordinates are (-1.5, -0.5). The result obtained for this query is shown in Table.V.

Table V: Search Results

Feature size	Features selected	Search Radius in (Km)	Input co-ordinates	Output co-ordinates
2	Hotel, Park	20	15.6,-1.2	29.571394,7.4796944 34.901276,1.379443 20.061771,0.6429453 35.155598,5.203124
3	ATM, Park, Shopping mall	7	-1.5,-0.5	1.2688353,0.8004315 3.7381945,-0.221075 6

VI. CONCLUSION

This paper introduces a new technique namely CONNEKT for performing searches involving more than one feature with respect to a user location. This finds application in context aware location-based services, wherein a user may want to query about the existence of one or more facilities from his current location. CONNEKT makes use of the mined co-located pattern instances which are readily available as the output of a co-location mining algorithm, which are mapped on to a K-d Tree for efficient retrieval.

The Co-located nearest neighbor search is efficient when compared to the searches involving join operations. The usage of mined Co-located pattern improves the querying time. In future, the same algorithm can be modified for LBS with preferences given to the network distance.

REFERENCES

1. S. Shekhar, P. Zhang, and Y. Huang, "Spatial Data Mining," in *Data Mining and Knowledge Discovery Handbook*, New York: Springer-Verlag, 2010, pp. 833–851.
2. M. Camilli, "Preserving Co-Location Privacy in Geo-Social Networks," 2018.
3. C. Hongfei and W. Xiaoyan, "Research on GIS-based spatial data mining," *2011 Int. Conf. Bus. Manag. Electron. Inf.*, vol. 1, pp. 351–354, 2011.
4. G. T. S. J. F. Yu Y.L., "Spatial data mining based on campus GIS," *Adv. Mater. Res.*, vol. 282–283, pp. 641–645, 2011.
5. L. Zhao, X. Zheng, and S. Wang, "Design and Implementation of Spatial Data Mining System (M-SDM) based on MATLAB," *Journal of Computers*, vol. 3, no. 10. 2008.
6. G. Zhou, R. Zhang, and D. Zhang, "Manifold Learning Co-Location Decision Tree for Remotely Sensed Imagery Classification," *Remote Sens.*, vol. 8, no. 10, p. 855, Oct. 2016.
7. Y. Dong and G. Li, "Mobile application for hydrogeologic field investigations," *Environ. Model. Softw.*, vol. 53, pp. 62–64, 2014.
8. N. Jean, M. Burke, M. Xie, W. M. Davis, D. B. Lobell, and S. Ermon, "Combining satellite imagery and machine learning to predict poverty.," *Science*, vol. 353, no. 6301, pp. 790–4, 2016.
9. H. Wang, D. Kifer, C. Graif, and Z. Li, "Crime Rate Inference with Big Data," 2016.
10. W. Chen, F. Guo, and F. Wang, "A Survey of Traffic Data Visualization," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 6, pp. 2970–2984, 2015.
11. L. Wang, J. Han, H. Chen, and J. Lu, "Top- k probabilistic prevalent co-location mining in spatially uncertain data sets," vol. 10, no. 3, pp. 488–503, 2016.
12. W. Yu, T. Ai, Y. He, and S. Shao, "Spatial co-location pattern mining of facility points-of-interest improved by network neighborhood and distance decay effects," *Int. J. Geogr. Inf. Sci.*, vol. 8816, no. July, pp. 1–17, Jun. 2016.
13. X. Yao, L. Chen, L. Peng, and T. Chi, "A co-location pattern-mining algorithm with a density-weighted distance thresholding consideration," *Inf. Sci. (Ny).*, vol. 396, pp. 144–161, 2017.
14. G. Adomavicius and A. Tuzhilin, "Context-aware recommender systems," in *Recommender Systems Handbook, Second Edition*, 2015, pp. 191–226.
15. L. Hu, S. Nooshabadi, and M. Ahmadi, "Massively parallel KD-tree construction and nearest neighbor search algorithms," *2015 IEEE Int. Symp. Circuits Syst.*, pp. 2752–2755, 2015.
16. W. Yu, "Spatial co-location pattern mining for location-based services in road networks," *Expert Syst. Appl.*, vol. 46, pp. 324–335, Mar. 2016.
17. D. Parsons, A. Rashid, A. Telea, and A. Speck, "An architectural pattern for designing component-based application frameworks," no. May 2003, pp. 157–190, 2006.
18. X. Bai, X. Dong, and Y. Su, "Edge propagation KD-trees: Computing approximate nearest neighbor fields," *IEEE Signal Process. Lett.*, vol. 22, no. 12, pp. 2209–2213, 2015.
19. M. Kuo, L. Chen, and C. Liang, "Building and evaluating a location-based service recommendation system with a preference adjustment mechanism," *Expert Syst. Appl.*, vol. 36, no. 2, pp. 3543–3554, 2009.
20. [20] M. Sheshikala, D. R. Rao, and R. V. Prakash, "Parallel Approach for Finding Co-location Pattern – A Map Reduce Framework," *Procedia Comput. Sci.*, vol. 89, pp. 341–348, 2016.
21. J. L. Bentley, "Multidimensional Binary Search Trees Used for Associative Searching," 1975.
22. X. Chuanjie, L. Gaohuan, G. Bingbo, and S. Wentao, "The optimization of remote spatial join queries on spatial information grid," *2009 WRI World Congr. Comput. Sci. Inf. Eng. CSIE 2009*, vol. 4, pp. 356–359, 2009.
23. K. Xiao, X. S. Hu, B. Zhou, and D. Z. Chen, "Shell: A Spatial Decomposition Data Structure for Ray Traversal on GPU," *IEEE Trans. Comput.*, vol. 65, no. 1, pp. 230–243, 2016.
24. N. Bereczky, A. Duch, K. Németh, and S. Roura, "Quad-kd trees: A general framework for kd trees and quad trees," *Theor. Comput. Sci.*, vol. 616, pp. 126–140, 2016.
25. D. Liu, "Efficient k nearest neighbor queries on remote spatial databases using range estimation," *Proc. 14th Int. Conf. Sci. Stat. Database Manag.*, pp. 121–130, 2002.
26. W. C.-M. Liaw, Yi-Ching, Leou Maw-Lin, "Fast exact k nearest neighbors search using anorthogonal search tree," *Pattern Recognit.*, vol. 43, no. 6, pp. 2351–2358, 2010.
27. W. Liao, Z. Zhang, Z. Yuan, W. Fu, and X. Wu, "Parallel continuous k-nearest neighbor computing in location based spatial networks on GPUs," *Proc. - 2013 Int. Conf. Comput. Inf. Sci. ICCIS 2013*, pp. 271–274, 2013.
28. C. M. Eastman and M. Zemankova, "Partially specified nearest neighbor searches using k-d trees," *Inf. Process. Lett.*, vol. 15, no. 2, pp. 53–56, 1982.
29. P.-Y. Po-YiLi, W.-C. P. W.-C. Peng, T.-W. W. T.-W. Wang, W.-S. K. W.-S. Ku, I. X. I. Xu, and J. a. Hamilton, "A Cloaking Algorithm Based on Spatial Networks for Location Privacy," *2008 IEEE Int. Conf. Sens. Networks, Ubiquitous, Trust. Comput. (sutc 2008)*, pp. 90–97, 2008.
30. N. Kumar and S. Siva Sathya, "Clustering assisted co-location pattern mining for spatial data," *Int. J. Appl. Eng. Res.*, vol. 11, no. 2, pp. 1386–1393, 2016.

AUTHORS PROFILE

Sharmiladevi is a Ph.D. Scholar in the Department of Computer Science, School of Engineering & Technology, Pondicherry University. She has completed her Master degree (M. Tech.) in computer science and engineering and is currently pursuing Ph.D. degree. Her research interests includes Data mining, Spatial Data Mining.

Dr. S. Siva Sathya is a Professor in the Department of Computer Science, Pondicherry University. She completed her M.Tech & Ph.D. in Computer Science & Engineering in the Pondicherry University and subsequently joined as a faculty. Her research interests include Bio-inspired computing, Data Mining, Vanet, etc. She has to her credit several research papers in International Journal and Conference. She has also received the "Naari Shakthi Puraskar 2017" from the Hon.President of India for the women safety app MITRA.

Dr. Naveen Kumar is an Assistant Professor in the Department of Computer Science and Engineering, KL University, Vaddeswaram, Guntur-522502. He has completed his MCA & Ph.D in the Department of Computer Science & Engineering, School of Engineering & Technology, Pondicherry University, Pondicherry. His research area includes Data Mining and Spatial Data Mining.