A Framework for Geographical based Approximate String Search

G. Vidya Praveena[1], K. Krishna reddy[2], Janapati Venkata Krishna [3]

[1]pursuing M.Tech (CSE), [2]Assistant Professor (CSE Department), [3]Associate Professor & HOD (CSE Department)

[1,2,3]Holy Mary Institute of Technology and Science, Keesara, Affiliated to JNTU- Hyderabad, A.P, India

***Abstract*— This work deals with the approximate string search in large spatial databases. We investigate range queries augmented with a string similarity search predicate in both Euclidean space and road networks. We make this query the spatial approximate Existing (Apr 19, 2013) string (SAS) query. In Euclidean space, we propose an approximate solution MHR-tree, which will embed min-wise signatures into R-tree. These min-wise signature for an index node u keeps a concise representation of the union of q-grams from strings under the sub-tree of u. We here analyze the pruning functionality of such signatures based on the set resemblance between he query string and the q-grams from the sub-trees with index nodes. Here we also discuss how to estimate the selectivity of a SAS query in Euclidean space, for which here we present a novel adaptive algorithm to find balanced partitions using both the spatial and string information stored in the tree as discussed. For queries on road networks, we propose a novel method, the RSASSOL, which significantly outperforms the baseline algorithm in practice. RSASSOL combines the q-gram based inverted lists and the pruning based on reference nodes. Extensive experiments on large real data sets demonstrate the efficiency and effectiveness of the approach.**

I. INTRODUCTION

The Keyword search in a very large amount of data is important operation in almost every domain which deals with collection or storing of the data. When we extend this study of databases to Spatial Data Bases, the keyword search will become the most fundamental building block for the increasing amount of the real world applications. But most of the present day's keyword based searches such as IR^2 –Tree will retrieve only exact keyword matches and not all. Since this type of exact match of keyword is a special case of approximate String matching it is very clear that when exact keyword search may lead us to a situation where the exact data required may not be found since we do not know what is the exact key word associated with that data. So the approximate string search is important when to uses have a fuzzy key search condition, or when a spelling mistake may occur when submitting the query. In this concept of approximate string search this can be mixed and used with any combination of spatial queries. Here let us make that the keyword you have given acts as a point and the required data as a line. Here when it is regarding with the exact string search mostly it may connect too only on line which exactly matches whatever the keyword that is given but while coming to the approximate string search it may not link only to one line but to too many lines. Here it means it will take all the possible data which will have the approximation to the keyword that means which have may be around more than half matched with the keyword entered. A straightforward solution to the Spatial Approximate String query is to use the existing techniques for getting the spatial component of Spatial Approximate String Search query and to verify the approximate string match either in post-processing or on the intermediate results of the spatial search.

The main contribution of this paper are as follows:

- Formalizing the notion of Spatial Approximate String queries and related selective estimation problem
- Here we present a robust and novel selectivity estimator. Our idea is to make an adaptive algorithm that can find balanced partitions from any tree based index based on both the spatial and string information nodes.
- The RASSSOL partitions the road network, searches the relevant sub graphs, and then prunes candidate points using both the spatial reference nodes and string matching index.
- We here demonstrate the effectiveness and efficiency of proposed methods for SAS queries by using a comprehensive experimental evaluation.

PROBLEM FORMULATION:

Keyword search over a large amount of data is an important operation in all the domains that include storing and retrieval of data by searching. The recently extended study of spatial databases from common databases, where keyword search becomes a fundamental building block for an increasing number of real-world applications, and proposed the IR -Tree. A main limitation of the IR -Tree is that it only supports exact keyword search. So when the exact keyword

search is being employed always there will a condition where the user cannot get the required data that is because of the situation may be he may have spelt it wrong or in some cases may be he do not know the exact keyword that is associated with the data that he wanted to access, so as long as he do not know the exact key word he will not be able to get the data.

The main drawback for this is always the requirement of the exact keyword for getting the data which may not be always known.

PROPOSED SOLUTION:

For RSAS type queries where string match approximate is enough, the baseline of the spatial solution is based on the Dijkstra's algorithm. Take the given query as point q, and the query range radius as r, and a string, we now expand from query point q on the road network using Dijkstra algorithm until we reach all the possible points that are distance r away from q and then verify the string predicate either in post-processing step or on the intermediate results of expansion. We now denote this approach as the Dijkstra solution. Its performance will degrade quickly when the query range enlarges or when the data on the network increases or in both cases. This motivated us to find a novel method that can avoid the unnecessary expansions of road network, by combining the pruning from both the spatial and the string predicates simultaneously.

We here demonstrate the effectiveness and efficiency of our proposed methods for Spatial Approximate String queries which are using a comprehensive experimental evaluation for the results. For ESAS queries, our experimental evaluation is able to cover both the synthetic and real data sets of up to 10 million points and in 6 dimensions on the other hand for RSAS queries, our evaluation is based on two large and real road network datasets that contain nearly 175,800 nodes, 179,100 edges, and 2 million points on the road network. In both the cases, our method significantly outperformed the respective baseline methods.

The most explained advantage of the proposed solution is the user will get the exact data that he was searching for even though the keyword he entered is not perfect. Here I also gives one more feasibility to the user that there is no need for him to remember the exact keyword or to be exact with the spelling of the keyword he is entering there.

3 THE RSAS QUERIES:

In case of RSAS Queries, since the locations of points which are constrained by the network and will be represented by the edge holding the point and the distance which acts as the offset to the edge end. To handle large scale datasets, we here develop eternal-memory algorithms and adapt a disk-based road network storage framework.

3.1 Disk-based road network representation

Here we adopt a disk-based storage model to the setting that will group the network nodes based upon their connectivity and distance. The following figure demonstrates an instance of model proposed for the network. In the model of ours, the adjacency list and points will be stored in two different files, each of them is indexed by a separate V_R of the nodes form V is selected for reference nodes. The distance between any two nodes or two points or a node from a point will make the length of the shortest path.

For the node n_i, its distance from reference nodes in V_R and number of nodes of n_i. For every adjacent node n_j of n_i. we will store the adjacent node's ID, the length of the edge $e=(n_i,n_j)$ and a pointer that points group on e. See the following figure,

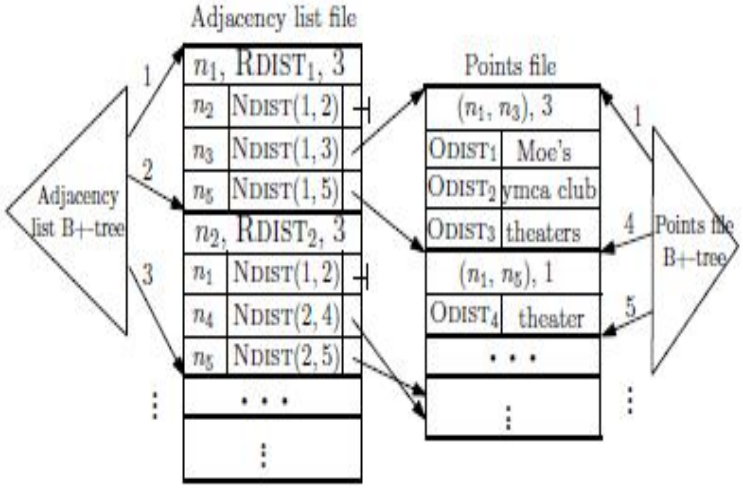

Fig. 1 Disk- based storage of the road Network

In the described points file, the ids of points will be assigned in such a way that for points of the same edge (ni, nj), points will be stored by their own offset distances to the a node with smaller id in arranged in ascending order, and then their ids are then sequentially assigned (crossing different edges as well). Note that any edge e defined with two nodes ni and nj , we will represent e by always placing that node with the smaller id first. That is, if ni < nj, in the adjacency list of nj , the entry for ni will have pointer that to the points group which is pointing to the points group of (ni, nj) (i.e., no duplication of any points group will be stored). We will also store other information that is associated with a point (i.e., strings) after the offset distance. We now store the points on the same edge in points group. At the inception of the points group, we have to store the edge information (i.e., (ni, nj)) and the also the number of points on the edge. The groups are supposed to be stored in a points file in ascending order of the node ids that are defining the edges. Then B+-tree will be built on this file with the keys being the first point id of each points group and values that are being in the corresponding points group. For example, if the points file in Figure 1 partially reflect the example. The ODISTi was the offset distance of point pi.

3.2 The RSASSOL algorithm:

We partition a road network G={V,E} into m edge-disjoint sub graphs. We also select one small subset of nodes as reference nodes. Given below is the overview of RSASSOL algorithm. Which by concept have five steps. When a query is given, first find all the sub graphs that are intersecting with query range. Then Filter trees of these sub graphs to retrieve the points which have the strings that are potentially similar to the query string.

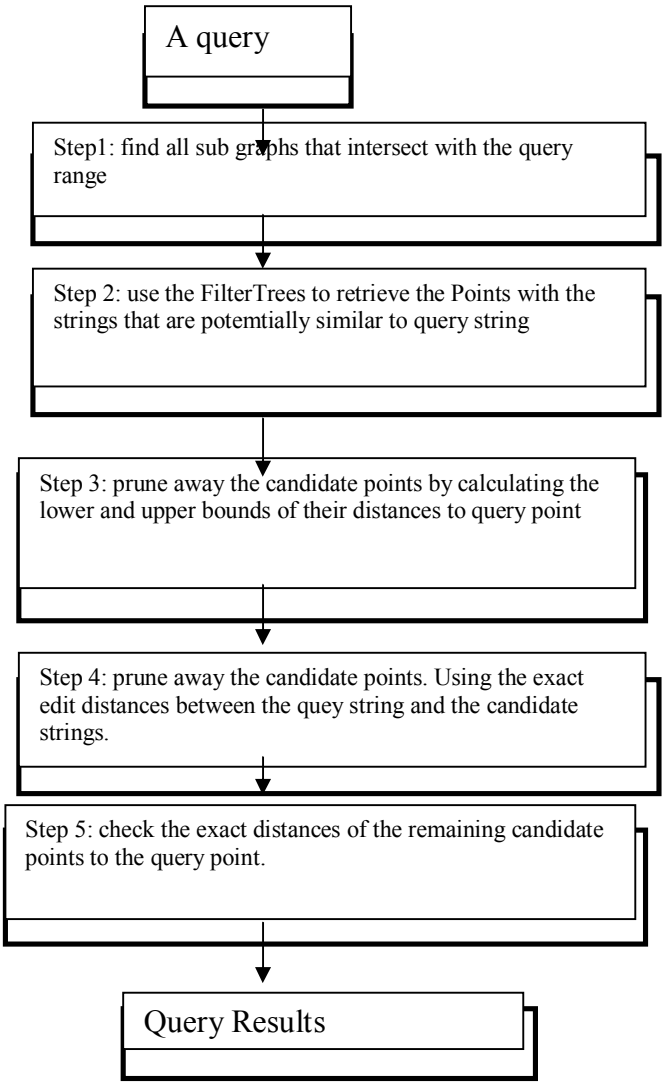

Fig 2: overview of RSASSOL ALGORITHM

Algorithmic Explanation of above discussed over view is as fllows.

Algorithm : RSASSOL(network G, Q_r = (q, r), Q_s = (_, _))

/* *step 1: find sub graphs intersecting* **1** *the query range* */
2 Find the set X of ids of all sub graphs intersecting (q, r);
3 Set A = ∅, Ac = ∅;
4 for *each sub graph id* i ∈ X **do**
5 /* *step 2: use the Filter Trees to retrieve points with strings that are potentially similar to the query string* */
6 Find all point ids in Gi whose associated strings σ′ may satisfy ε(σ′, σ) ≤ τ using Filter Treei, and insert them into Ac;
7 /* *step 3: prune away points by calculating the lower and upper bounds of their distances to the query point using* VR */
8 for *every point* pi ∈ Ac **do**
calculate d+**9** (pi, q) and d−(pi, q) as discussed;
if d+**10** (pi, q) ≤ r **then**
11 if ε(σi, σ) ≤ τ **then**
12 move pi from Ac to A;
13 /* *step 4: prune points using the exact edit distance between the query string and the candidate string* */
14 else
15 delete pi from Ac;
16 else
17 if d−(pi, q) > r **then**
18 delete pi from Ac;
19 /* *step 4: same pruning as the step 4 above* */
20 for *every point* pi ∈ Ac **do**
21 if ε(σi, σ) > τ **then**
22 delete pi from Ac;
23 /* *step 5: check the exact distances of the remaining candidate points to the query point* */
24 Use the MPALT algorithm to find all points p's in Ac such that d(p, q) ≤ r, push them to A;
25 Return A.

Query Processing: RSASSOL algorithm is presented as above described. First, we will find all possible sub graphs that intersect with query range. We here employ the Dijkstar's algorithm for traversing the nodes in G, starting from query point q. Whenever traversal meets the first node of new sub graph, we check the sub graph for further exploration. This algorithm terminates when the boundary of the query range is reached. For each Sub graph to examine it we give approximate string to search over Filter tree as the next pruning step to be taken.
Further prune the points using the spatial predicate, by computing lower and upper bounds. Recall that we have pre computed the distance of every network node from every node.

4.3 Selectivity estimation of RSAS queries

The Selectivity estimation of range queries on the road network is a very harder problem than its counter part of Eiclidean Space. Many methods are proposed for this but they are only able of estimating the count of nodes and edges in range. Nothing can be efficiently adapted to estimate the number of points. One unsophisticated solution is to treat

points as nodes in the network which can be achieved by introducing more number of edges. This will certainly going to increase the space consumption of the network as always the number of points will be greater than the number of nodes existing in the network sometimes this difference between the count of number of nodes and the number of points will be very large. This will also leave a problem that integrating string selectivity estimator with the spatial selectivity estimator as we did for Euclidian space.

EXPERIMENTAL EVALUATION:

For the ESAS queries we have implemented R-tree solution and the MHR-tree using the widely adopted spatial library. The Adaptive R tree algorithm seamlessly for ESAS selectivity estimator for both R-Tree and the MHR tree. For RSAS queries, we implemented the Dijkstra algorithm as solution and the RASSOL method, based upon the disk-based storage model and Flamingo package tree.

Setup: The spatial predicate r(a rectangle) is an ESAS query and is generated by selecting a center pint c_r and a query area that is specified as percentage of total space, denoted by θ = area of r/θ(p) randomly. For making sure that query should return non-empty results, we here select the query string as associated String of nearest neighbor center point from P. The default value of θ is 3%. Default size of signature is l = 50 hash values. The spatial range predicate for a RSAS query can be generated by choosing point q on randomly selected edge and radius r is network distance value.

RELATED WORK:

IR^2-tree was proposed to perform exact keyword search with kNN queries of spatial databases. The IR^2 – tree cannot support spatial approximate string searches, authors in study of the m-closet type keywords query in Euclidian space, where proposed cannot handle the approximate string search neither.

Another related work appears where LBAK-tree was proposed to answer location-based approximate search keyword queries which look similar to our definition of spatial approximate string queries and their working in Euclidian Space. The basic idea of this methodology is tree based spatial augmentation. With the q—grams sub tree nodes for supporting the edit distance based approximate string/keyword search. LBAK-tree was proposed after study on spatial approximate string queries in the Eucllidian space. The results of these shows that LBAK-tree has successfully achieved better query time than the MHR-tree.

Best of our knowledge, RSAS queries and the selectivity estimation of the spatial approximate String queries have not been explored before. The approximate string search alone has been an extensively studied thing in the literature. All the works done on this generally assume a similarity function which will quantify the closeness between the two strings on the entered keyword and the other is the search keyword. A variety of these functions like edit distance are there. Many of the approaches leverage the concept about q-terms.
Note that the concept of Approximate string matching will also some times refers to a problem of finding a pattern string approximately in a text. The problem explained in our paper is different: we want to search in a collection of strings which serve as keywords for a particular data and now to find those similar to a single query string.

CONCLUSION:

In this paper presentation of comprehensive study on spatial approximate string search is being explained in both Euclidean space and road Networks. Here we use the edit distance as similarity measurement of a string predicate. Here we also address problem of selectivity estimation of queries in Euclidian space. The Future Work may include examining spatial approximate sub-string, making methods that are more update-friendly and solving the problem of selectivity estimation for RSAS queries.

REFERENCES:

[1] S. Acharya, V. Poosala, and S. Ramaswamy. Selectivity estimation in
spatial databases. In *SIGMOD*, pages 13–24, 1999.
[2] S. Alsubaiee, A. Behm, and C. Li. Supporting location-based
approximate-keyword queries. In *GIS*, pages 61–70, 2010.
[3] A. Arasu, S. Chaudhuri, K. Ganjam, and R. Kaushik. Incorporating
string transformations in record matching. In *SIGMOD*, pages 1231–
1234, 2008.
[4] A. Arasu, V. Ganti, and R. Kaushik. Efficient exact set-similarity joins.
In *VLDB*, pages 918–929, 2006.
[5] N. Beckmann, H. P. Kriegel, R. Schneider, and B. Seeger. The R_-tree: an efficient and robust access method for points and rectangles. In
SIGMOD, pages 322–331, 1990.
[6] A. Z. Broder, M. Charikar, A. M. Frieze, and M. Mitzenmacher. Minwiseindependent permutations (extended abstract). In *STOC*, pages
327–336, 1998.
[7] X. Cao, G. Cong, and C. S. Jensen. Retrieving top-k prestige-basedrelevant spatial web objects. *Proc. VLDB Endow.*, 3:373–384, 2010.
[8] K. Chakrabarti, S. Chaudhuri, V. Ganti, and D. Xin. An efficient filterfor approximate membership checking. In *SIGMOD*, pages 805–818,
2008.

AUTHOR PROFILE

G.Vidya Praveena, pursuing M.Tech (CSE) from Holy Mary Institute of Technology and Science, Keesara, Ranga Reddy Dist., Affiliated to JNTU-HYDERABAD.

K.Krishna Reddy, **Assistant Professor** (CSE Department), Holy Mary Institute Of Technology and Science, Keesara, Ranga Reddy Dist., Affiliated to JNTU-HYDERABAD.

Janapati Venkata Krishna, **Associate Professor & H O D** (CSE Department), Holy Mary Institute Of Technology and Science, Keesara, Ranga Reddy Dist., Affiliated to JNTU-HYDERABAD.