IOSR Journal of Engineering (IOSRJEN) **www.iosrjen.org**
ISSN (e): 2250-3021, ISSN (p): 2278-8719
Vol. 08, Issue 9 (September. 2018), ||V (I) || PP 33-39

An Efficient and Improved Query Search in Road Networks

[1]A.DurgaPraveenKumar, [2]Dr.V.Suresh.[3]V.V.S.Sasank
[1,2,3]Asst.Professor, IT Department,ANITS EngineeringCollege,Visakhapatnam

Abstract: Allowing the user to search from road networks is an interesting research issue in the field of queries in road network. Various models proposed by various authors from years of research, every model has its own pros and cons. If this paper, we are proposing an efficient and empirical model of road network based implementation for user search queries. Finding the search results like restaurant, hospital, home etc. are related spatial queries which includes latitude and longitude for every entity. Our proposed model improves the performance by cluster and proxy implementation while finding the optimal path and it gives more efficient results than traditional model.

Date of Submission: 26-08-2018 Date of acceptance: 07-09-2018

I. INTRODUCTION:

A few calculations have been created utilizing the network remove. Shahabi et al. proposed an inserting method to change a street network into a higher dimensional space so as to use computationally straightforward measurements [1][2]. The primary impediment of this method is that it gives just an estimate of the real separation. Jensen et al. [3] propose an information model and meaning of dynamic usefulness required for NN questions in spatial network databases. They utilize calculations like Dijkstra's calculation all together to perform online counts of the most brief separation from a query point to a protest. Shahabiet al. [4] display four elective systems for finding the first closest neighbour to a moving query question on guaranteed way.

In past works, introduced an answer called INE for NN inquiries in spatial network databases by introducing a design that incorporates network and Euclidean data and catches down to business constraints. Since the quantity of connections and hubs that should be recovered and inspected is contrarily professional portioned to the cardinality proportion of items, the primary inconvenience of this approach is a sensational degradation in execution when the above cardinality proportion is small, which is a typical case for genuine scenarios. Moreover, it doesn't improve crude musical dramations to encourage an effective network seek since it is intended to help both customary spatial inquiries in light of the Euclidean separation and inquiries in light of the network remove[5].

The instinct behind is that clients (e.g. suburbanites) want to take after the course they are natural with, in this way they might want to pick the corner store with the littlest deviation from the course. In the wake of refuelling, they will come back to the past course and proceed with the excursion. Anyway a downside of IRNN is that the client needs to enter precisely the entire query way ahead of time, which is recognized by all crossing points along the way, while a client's driving way frequently can't be unequivocally pre-chosen. Envision that a client is driving from Washington to New York, which is a long trip. It is unrealistic for a client to enter hundreds of convergences before effectively making a query.

The Nearest Neighbour query is also extended to a roadnetwork scenario by using network distance as the distancemetric. Papadis et al. present in [5] the Incremental EuclideanRestriction (IER) and Incremental Network Expansion(INE) algorithms for retrieving k-NN according to networkdistance. IER uses the Euclidean distance as a lowerbound for pruning during the search, and INE performs anetwork expansion similar to the Dijkstra's algorithm .Jensen et al. also propose in a general spatial-temporalframework for NN queries in a road network which is representedby a graph.

II. RELATED WORK:

In portable correspondence database and innovations are rising applications. There is a capacity to help and continuous inquiries from versatile customers in street authorizations to duplicate material is conceded given. The capacity to help consistent inquiries from portable customers on a street Permission to duplicate without charge all or part of this material is conceded given that the duplicates are not made or circulated for coordinate business advantage, the VLDB copyright see and the title of the production and its date show up, and see is given that replicating is by authorization of the Very Large Data Base Endowment [6][7].

Example arrangement frameworks in view of machine learning calculations are ordinarily utilized as a part of security-related applications like biometric validation, network interruption recognition, and spam sifting, to segregate between a "genuine" and a "malignant" example class (e.g., authentic and spam messages).

In spite of conventional ones, these applications have an inborn ill-disposed nature since the info information can be deliberately controlled by a savvy and versatile enemy to undermine classifier task. This frequently offers ascend to a weapons contest between the enemy and the classifier originator. Surely understood cases of assaults against design classifiers are: presenting a phony biometric quality to a biometric verification framework [8][9].

The is first seek after security with regards to a weapons contest it isn't adequate to respond to watched assaults, yet it is likewise important to proactively envision the enemy by foreseeing the most significant, potential assaults through an imagine a scenario where examination; this enables one to create reasonable countermeasures previously the assault really happens, as indicated by the guideline of security by outline. Second, to give pragmatic rules to reproducing reasonable assault situations, we characterize a general model of the foe, regarding her objective, information, and capacity, which envelops and sums up models proposed in past work. Third, since the nearness of painstakingly focused on assaults may influence the conveyance of preparing and testing information independently, we propose a model of the information circulation that can formally describe this conduct, and that enables us to consider an expansive number of potential assaults; we additionally propose a calculation for the age of preparing and testing sets to be utilized for security assessment, which can normally oblige application-particular and heuristic systems for recreating assaults[10].

III. GENERALIZED COSTS

In its original formulation [1, 7], the PRP problem consists of finding a path of minimum user-specified linear combination of additive costs. However, this is too restrictive in practice as some important constraints cannot be modeled as additive costs. For example, one cannot simply add height limitations of two consecutive tunnels. Other real-world restrictions such as vehicle width, vehicle weight, or maximum climbing ability (depending on the slope) essentially fall into the same category: Every road has a certain threshold value (i. e., the tunnel height), and if the vehicle's characteristic value (i. e., its height) is above this threshold, the vehicle is not allowed to traverse the road. Clearly, adding these threshold values is not meaningful, instead one needs to compute the minimum of thresholds: A vehicle can pass through every tunnel on a path, if and only if it can pass through the lowest tunnel. Restrictions that are formalized by upper bounds on vehicle characteristics are the most common. However, there also restrictions that result in a lower bound. An example is the minimum required speed on highways that bans vehicles that cannot go fast enough. Another source of restrictions is that some road categories are forbidden for some vehicle types. For example many city centers ban large trucks. Some trucks carry dangerous goods and are therefore not allowed in water conservation zones. Some drivers want to avoid highways with toll. All of these restrictions have in common that some roads are flagged and some vehicles are not allowed to traverse them. It is possible to regard them as 1-bit height-limitations. However, we prefer another view: We attach to every road a bitfield where the i-th bit stands for the i-th restriction of this type. By convention we say that a bit being set means that a road can be traversed. A path can be traversed if every road in it can be traversed. Formally this consists of computing the bitwise-and of all road bitfields and testing the bits in the result against the vehicle restrictions or user preferences. We support all these criteria by generalizing the PRP scenario. The user does not input a vector of query weights w, but an arbitrary function f that fulfills a set of requirements. We require f to map cost vectors onto a value from $R \geq 0 \cup \{\infty\}$. We further need an operation $\circ$ that combines two cost vectors. We require that $\circ$ is associative, i.e., for any cost vectors c1, c2, and c3 we require that $(c1 \circ c2) \circ c3 = c1 \circ (c2 \circ c3)$. Furthermore, it must not matter whether we first combine two cost vectors c1 and c2 and then apply f, or whether we first apply f to both vectors and then compute the sum of the results. Formally, we require that $f(c1 \circ c2) = f(c1) + f(c2)$, which is the definition of f being a semigroup homomorphism. In the case of linear combinations, f is the scalar product with w, and the $\circ$-operation is the component-wise addition. However, we can also do component-wise minimum or maximum, since it is associative, and even choose different operations for different cost components. The right operation for height limitations (and similar restrictions), is to compute the minimum of all height limitations. The function f then maps the cost vector onto ∞ if the vehicle is too high and otherwise ignores that cost component. The $\circ$ operation for road categories is the bitwise-and operation, which is fortunately also associative. The function f tests whether a certain bit, such as the highway bit, is set or not. Depending on the outcome f evaluates to ∞ or f looks only at the other cost components.

IV. ACCELERATING TRAFFIC ASSIGNMENTS BY FAST BATCHED SHORTEST PATHS

Previous work applying speedup techniques to traffic assignment observed that the performance bottleneck depends on the traffic scenario under study. For short or off-peak periods, where there are few OD-pairs, preprocessing dominates the total running time. When there are many OD-pairs, as for long or peak periods, queries become the main bottleneck.

To decrease the preprocessing time, we apply the concept of customization to traffic assignment. Customizable speedup techniques [8, 2, 4] split preprocessing into a metricindependent part, taking only the graph structure into account, and a metric-dependent part (the *customization*), incorporating new edge weights

(the *metric*). Since the graph topology does not change in all iterations of the traffic assignment procedure and only edge weights change, it suffices to run a fast customization in each iteration instead of an entire preprocessing. We build our accelerated traffic assignment upon customizable contraction hierarchies [1, 2], which allows us to employ the hierarchy decomposition optimization from [2, 6]. As basic query algorithm, we use the engineered elimination tree search from the previous section. To reduce the query time, the following sections introduce several optimization techniques for computing batched point-to-point shortest paths fast.

4.1 Reordering OD-pairs to Exploit Locality

Previous work processed the OD-pairs in no particular order. However, reordering the OD-pairs so that pairs with similar forward and reverse search spaces tend to be processed in succession improves memory locality and cache efficiency. We call two search spaces similar if their symmetric difference is small. Note that the size of the symmetric difference between the search spaces of u and v is equal to the distance between u and v in the elimination tree. Hence, we partition the elimination tree into as few cells with bounded diameter as possible, assign IDs to cells according to the order in which they are reached during a DFS [29] on the elimination tree, and reorder OD-pairs lexicographically by the origin and destination cells.

We use a simple yet optimal greedy algorithm to partition the elimination tree into as few cells with diameter at most U as possible. Our algorithm repeatedly cuts out a subtree (with diameter at most U) and makes it a cell of its own. In order to do so, it maintains for each vertex v the height $h(v)$ of the remaining subtree$T_v$rooted at v, initialized to zero, and processes vertices in ascending rank order. To process v, we examine its children $w_i$in order of increasing height of T_{wi}. If $h(v) + 1 + h(w_i) \leq U$, we set $h(v) = 1 + h(w_i)$. Otherwise, we cut out T_{wi}, making it a cell of its own. We use $U = 40$ in our experiments.

4.2 Centralized Searches

Instead of processing similar OD-pairs *in succession*, processing them *at once* in a single search achieves additional speedup. The idea of bundling together multiple shortest-path computations was introduced in [20] and later used in [8, 7, 9, 4, 40]. However, in each case, centralized searches were only used for one-to-all and -many queries, and only combined with plain Dijkstra (and Bellman-Ford in [8]). Even (R)PHAST [7, 9] performs the CH searches sequentially, and bundles only the linear sweeps. We extend the idea to point-to-point queries, and combine it with CH searches, including appropriate stopping and pruning criteria.

The basic idea of centralized searches is to maintain k distance labels for each vertex u, laid out consecutively in memory. The i-th distance label represents the tentative distance from the i-th source to u. Initially, the i-th distance label of the i-th source is set to zero, and all remaining $kn - k$ distance labels to ∞. When relaxing an edge (u,v), we try to improve all k distance labels of v at once. Increasing k allows us to compute more shortest paths at once, however, it also evicts useful data from caches.

Dijkstra-Based Search. Initially, we insert all k sources (targets) into the queue of the forward (reverse) search. As keys, we can use many different values, for example the minimum over all k entries in a distance label or the minimum over the entries that were improved by the last edge relaxation. However, a preliminary experiment showed that using the minimum over *all* k entries clearly dominates the others, which is consistent with previous observations on related techniques [20]. We can stop the forward (reverse) search as soon as its queue is empty or the smallest queue entry exceeds the maximum over all k tentative shortest-path distances. When using stall-on-demand [18], we prune the forward (reverse) search at a vertex v when each of the k distance labels of v is suboptimal.

Elimination Tree Search. Computing multiple shortest paths in a single elimination tree search is more involved, since it uses no queues that can easily be initialized with multiple sources and targets. Instead, we equip the forward and reverse search each with a tournament tree [23]. Suppose we have k sorted sequences that are to be merged into a single output sequence, as in k-way mergesort. To do so, we have to repeatedly find the smallest from the leading elements in the k sequences. This can be done efficiently with a tournament tree.

In our case, the k sorted sequences are the paths in the elimination tree T from each source (target) to the root, and the single output sequence is the order in which we want to process the vertices during the search. More precisely, we initialize the tournament tree with all k sources (targets). Then, we extract a lowest-ranked vertex from the tournament tree, process it, and insert its parent in T into the tournament tree. We continue with a next-lowest-ranked vertex, until we reach the root of T. Note that in our case, unlike in mergesort, the sequences are implicit, and never stored explicitly.

As soon as two (or more) of the k paths in T converge at a common vertex, there are duplicates in the single output sequence. However, we want to process each vertex at most once. Therefore, whenever two or more paths converge, we block all but one of them, so that only one continues to move through the tournament tree. To do so, we insert for each path to be blocked a vertex with infinite rank into the tournament tree (instead of the next vertex on the path). We know that some paths converged, when we extract the very same vertex several times in succession from the tournament tree.

A clear advantage of the centralized elimination tree search is that it retains the *labelsetting*property, i.e., each vertex and each edge is processed at most once. In contrast, the centralized Dijkstra-based search is a *label-correcting* algorithm. Note that one centralized elimination tree search is slower than k elimination tree searches by a factor of log$k$in O-notation (due to k-way merging), but outperforms them in practice (see Section 5).

4.3 Instruction-Level Parallelism

Modern CPUs have special registers and instructions that enable single-instruction multipledata (SIMD) computations performing basic operations (e.g., additions, subtractions, shifts, compares, and data conversions) on multiple data items simultaneously [24]. We implemented versions of the centralized searches using SSE instructions (working with 128-bit registers), and additionally versions using AVX(2) instructions (manipulating 256-bit registers), requiring a processor based on Intel's Haswell or AMD's Excavator microarchitecture.

As an example, we describe how an AVX-accelerated edge relaxation (used in Dijkstrabased and elimination tree searches) works, assuming $k = 8$. Since we use 32-bit distance labels, all k labels of a vertex fit in a single 256-bit register. To relax an edge (u,v), we copy all k distance labels of u to an AVX register, and broadcast the edge weight to all elements of another register. Then, we add both registers with a single instruction, and check with an AVX comparison whether any tentative distance improves the corresponding distance label of v. If so, we compute the packed minimum of the tentative distances and v's distance labels. In the same fashion, we implement the other aspects (stopping and pruning criteria).

4.4 Core-Level Parallelism

Dibbelt et al. [12] introduce parallelization techniques for the triangle enumeration during customization. However, we observed that the perfect witness search building the upward and downward search graphs (which is difficult to parallelize) actually takes up 60% of the customization time. Hence, the speedup obtainable by parallelizing the customization phase is limited (less than a factor of 1.5). For simplicity, we stick to sequential customization.

In contrast, the shortest-path computations are easy to parallelize. Since the centralized computations are independent from one another, we can assign contiguous subsets of OD-pairs to distinct cores. We distribute the OD-pairs to cores in chunks of size 64. This maintains some locality even between centralized computations. Each core executes a chunk, then requesting another chunk until no chunk remains. Flow units on the (shortcut) edges are cumulated locally and aggregated after computing all paths. We observe an almost perfect speedup for the time spent on queries.

V. COLLECTIVE SPATIAL KEYWORD QUERYING

With the proliferation of geo-positioning and geotagging, spatial web objects that possess both a geographical location and a textual description are gaining in prevalence, and spatial keyword queries that exploit both location and textual description are gaining in prominence. However, the queries studied so far generally focus on finding individual objects that each satisfy a query rather than finding groups of objects where the objects in a group collectively satisfy a query. We define the problem of retrieving a group of spatial web objects such that the group's keywords cover the query's keywords and such that objects are nearest to the query location and have the lowest inter-object distances. Specifically, we study two variants of this problem, both of which are NP-complete. We devise exact solutions as well as approximate solutions with provable approximation bounds to the problems. We present empirical studies that offer insight into the efficiency and accuracy of the solutions. With the proliferation of geo-positioning, e.g., by means of GPS or systems that exploit the wireless communication infrastructure, accurate user location is increasingly available. Similarly, increasing numbers of objects are available on the web that has an associated geographical location and textual description. Such spatial web objects include stores, tourist attractions, restaurants, hotels, and businesses. This development gives prominence to spatial keyword queries. A typical such query takes a location and a set of keywords as arguments and returns the single spatial web object that best matches these arguments. We observe that user needs may exist that are not easily satisfied by a single object, but where groups of objects may combine to meet the user needs. Put differently, the objects in a group collectively meet the user needs. For example, a tourist may have particular shopping, dining, and accommodation needs that may best be met by several spatial web objects. As another example, a user may wish to set up a project consortium of partners within a certain spatial proximity that combine to offer thecapabilities required for the successful execution of the project. To address the need for such collective answers to spatial keyword queries, we assume a database of spatial web objects and then consider the problem of how to retrieve a group of spatial objects that collectively meet the user's needs given as location and a set of keywords: 1) the textual description of the group of objects must cover the query keywords, 2) the objects are close to the query point, and 3) the objects in the group are close to each other. We present the new problem of retrieving a group of spatial objects, each associated with a

set of keywords, such that the group covers the query's keywords and has the lowest cost measured by their distance to the query point, and the distances between the objects in the group. We study two particular instances of the problem, both of which are NP-complete. We develop approximation algorithms with provable approximation bounds and exact algorithms to solve the two problems. Results of experimental evaluation offer insight into the efficiency and the accuracy of the approximation algorithms, and the efficiency of the exact algorithms.

VI. PROPOSED WORK:

In this paper we are proposing an empirical model query search implementation in road networks with cluster based hybrid implementation for shortest query result. Usually end user makes a query to search engine of the network. User query includes, search information along with location parameters. Road network or algorithm which implemented receives the input parameters, send those parameters to server to compute the available nodes which are nearest to the current location and filter the result and returns to the user. The implementation majorly divided in to two categories. One will cluster the available nodes and other filter he results with required information.

Intermediate Server or Sub Road Network receives set of nodes and cluster them based on the latitude and longitude with centroid based cluster implementation. Usually Many nodes exists in various spatial locations can be based on the latitude and longitude of the nodes. Server nodes can have grouped or clustered with geographic parameters and shortest distance can be computed with Euclidean distance as measure and keeps the nearest every time and repeats the same process until it is stabilizing or maximum number of iterations

Spatial Query Location based Cluster Implementation:

Step1:Load all available nodes from the network

Step2:Select number n of centroids from the complete set of nodes (N) and N>=n

Step3:Measure the Euclidean distance between C_i node and $N_i$node

Step4: For each N_i in N

If (Euclidean distance(C_i,N_j) <=initial distance) then

Shortest or optimal distance:= Euclidean distance(C_i,N_j)

Centroid_id=Ci;

End if

Next

Step 5: Continue the process until clusters are stable or maximum number of iterationsif not met.

Step6.:Continue the process or steps from 2 to 5

These clustered results can be forwarded to main server to process the information of nodes which are received from the intermediate server. It initially searches for complete object results and then sort the results based on the nearest distance between the required object and selected cluster object. It obviously improves the performance by minimizing the number of nodes

Location Query Search implementation:

Input: Qi—Input Spatial Query, DO_{list}(Total Data objects)

Output: R_{list} (result set)

1. User provides the spatial query which involves the spatial object and feature
2. Load DO_{list} from database(LBS)
3. For i=0;i<List_Nodes ;i++

For each Object O in DO_{list}

If(O== Q_i.objectname)

Add to Object_List

Next

For each object O in Object_List

If (O.attribute==Q_i.attribute)

Add 'O' to R_{list}

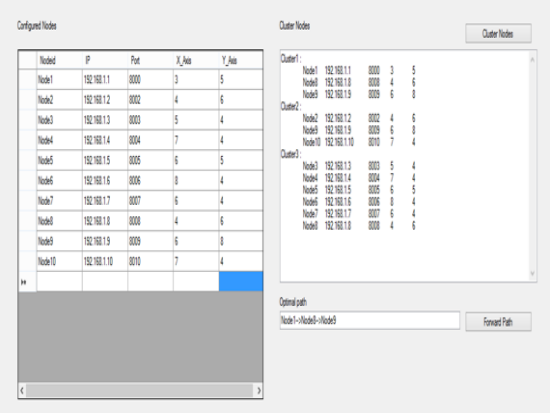
Next

Next

4. Sort the Result set
5. Return R_{list}

Output Analysis:

For Experimental analysis, we clustered the set of nodes based on the geographic or spatial information of the objects and results can be forwarded to intermediate server and process the spatial query. The following screen shows list of nodes with geo parameters andclustered nodes based on Euclidean distance between the nodes.



In the above implementation, it shows list of cluster nodes and shortest path generated by the server. Now main server identifies the object specific results and forwarded to end user as per input query

VII. CONCLUSION

In our work we are concluding our search implementation with a hybrid model. Usually end user makes a query to search engine of the network. User query includes, search information along with location parameters. Road network or algorithm which implemented receives the input parameters, send those parameters to server to compute the available nodes which are nearest to the current location and filter the result and returns to the user. Our proposed model gives more efficient results than traditional models.

REFERENCES:

[1]. D. Delling, A. V. Goldberg, and R. F. Werneck, "Faster batched shortest paths in road networks," in Proc. 11th Workshop Algorithmic Approaches Transportation Model., Optimization, Syst., 2011, vol. 20, pp. 52–63.

[2]. R. Geisberger, "Advanced route planning in transportation networks," Ph.D. dissertation, Karlsruhe Inst. Technol., Karlsruhe, Germany, Feb. 2011.

[3]. R. Geisberger, P. Sanders, D. Schultes, and C. Vetter, "Exact routing in large road networks using contraction hierarchies," Transportation Sci., vol. 46, no. 3, pp. 388–404, 2012.

[4]. I. Abraham, D. Delling, A. V. Goldberg, and R. F. Werneck, "A Hub-based labeling algorithm for shortest paths on road networks," in Proc. 10th Int. Symp. Exp. Algorithms, 2011, LNCS 6630, pp. 230–241.

[5]. I. Abraham, D. Delling, A. V. Goldberg, and R. F. Werneck, "Hierarchical hub labelings for shortest paths," in Proc. 20th Annu. Eur. Conf. Algorithms, LNCS 7501, 2012, pp. 24–35.

[6]. R. Bauer, D. Delling, P. Sanders, D. Schieferdecker, D. Schultes, and D. Wagner, "Combining hierarchical and goal-directed speed-up techniques for Dijkstra's algorithm," ACM J. Exp. Algorithmics, vol. 15, no. 2.3, pp. 1–31, 2010.

[7]. D. Delling, A. V. Goldberg, T. Pajor, and R. F. Werneck, "Customizable route planning," in Proc. 10th Int. Conf. Exp. Algorithms, LNCS 6630, 2011, pp. 376–387.

[8]. D. Delling, and R. F. Werneck, "Faster customization of road networks," in Proc. 12th Exp. Algorithms, LNCS 7933, 2013, pp. 30–42.

[9]. M. Holzer, F. Schulz, and D. Wagner, "Engineering multi-level overlay graphs for shortest-path queries," ACM J. Exp. Algorithms, vol. 13, no. 2.5, pp. 1–26, Dec. 2008.

[10]. Y.-W. Huang, N. Jing, and E. A. Rundensteiner, "Effective graph clustering for path queries in digital maps," in Proc. Conf. Inf. Knowl. Manage., 1996, pp. 215–222.

Authors

A. Durga Praveen Kumar is working as Asst Professor Dept of IT in ANITS Engineering College, Visakhapatnam. His research Paper published in **ELSEVIER publications** with Title "Data Collection, Statistical Analysis and Machine Learning Studies of Cancer Dataset from North Costal Districts of AP, India"-2015. AnotherPaper published in IJETT publications with Title "Improved Entity based Object Oriented Framework for IDE-June-2017". One more Paper published in IJETT publications with Title "An Empirical Model Of Range -Aggregate Queries In Big Data Environment-August-2016".

Dr V. Suresh received B.E. degree from AMA College of Engg., Kancheepuram, University of Madras, Chennai, Tamilnadu, India. Received M.Tech degree from Andhra University, Visakhapatnam, A.P., India. Received Ph.D. degree from Andhra University, Visakhapatnam, A.P., India. Presently he is working as Assistant professor, department of IT, ANITS Engineering College, Visakhapatnam. He is a life member in CSI(00174296),MITE(M155602) and AMIE(AM0915397). He is having 24 years of Teaching and Industrial Experience. His research interest includes Natural Language Processing(NLP) Image Processing, BigData, Data Analytics and Machine Learning.

Mr.V.V.S.Sasank, received his B.tech and M.tech Degree from GITAM University,Visakhapatnam, AndhraPradesh, India. He got 1st rank in M.tech CST from GITAMUniversity. Presently he is working as Assistant Professor in the IT Department of ANITS Engineering College,Visakhapatnam since one year. He activelyparticipated in various workshops and seminars and presented papers related to information technology. His areas of interests are data mining, databasemanagement system and image processing.

|

1A.DurgaPraveenKumar " An Efficient and Improved Query Search in Road Networks. ” IOSR Journal of Engineering (IOSRJEN), vol. 08, no. 9, 2018, pp. 33-39.