# Fuzzy Based Route Navigation in Directed Spatial Networks over Cloud

**Dr. K.B. Priya Iyer**[1]
[1] MOP Vaishnav College for Women(Autonomous), Chennai, India

***Abstract****- With advancements in Wireless technologies, GIS, GPS and sensors, spatial route search is gaining importance under location based services. The existing skyline query processing methods are based on travel time and not on a future date. This paper provides future travel time prediction based advanced navigation solutions for user desired point of interest. A new spatial query called Skyline Geo-Calendar Query (SGQ) on time dependent road networks is proposed. The SGQ is a spatial route search skyline query that retrieves nearest point of interest of the users for future journey schedule. The future traffic is predicted by considering factors such as day significance of the user spatial region, day light and climatic conditions of the journey date & time by applying fuzzy rules in MATLAB. The algorithm predicts the overall travel time in the user specified spatial region on the scheduled journey date. The behavior of the spatial query processing algorithms in terms of CPU time, network cost, node accessibility, and scalability of the dataset and the value of k is examined. The experimental simulation proves the accuracy of travel time prediction in real road networks and efficiency of algorithm in processing SGQ queries which provides best shortest path to reach point of interest.*

***Keywords***- Discrete event simulation, queuing system, size delay function

## I. INTRODUCTION

A Location Based Service (LBS) is an information retrieval System based on the user location provided by operators that is accessible by smartphones etc. LBS services utilize the geographic position of the mobile user device to provide location information to the consumer. Uses of LBS include mapping, navigation and social networking services based on location with technologies embedded either in the handset or placed in the network. LBS applications over smartphone include apps that identify a location of a person or object, such as finding the nearest ATM, Resturant, Hospital or the location of a friend or employee etc.

The integration of Global Positioning System (GPS) and Geographic Information System (GIS) technologies into Smartphone has given a whole new meaning to the concept of mobile phones, which earlier served the sole purpose of making and receiving calls and sending and receiving text messages. Finding routes is a popular application that provides real time traffic information and shortest path to road users, to increase their ability to choose the best alternative path.

In modern geographic information systems, navigation based queries represent an important class in spatial route search. For example, a skyline query can be tourists who wish to search for a high class hotel near-by a hospital and a beach. All the previous studies gives the route planning trips based on Euclidean distance, network distances instead of travel time.

The Skyline Geo-Calendar Query is finding the set of data objects by considering query to user specified location on a future journey schedule basing on travel time and climatic conditions. Travel time information plays an important role in a variety of real-time transportation applications. These real-time applications include advanced navigation information systems, Path Guidance Systems, and advanced Traffic Management Systems, Road Congestion management, Travel modeling and forecasting, Road network traffic simulation etc., which are part of the Intelligent Transportation Systems (ITS).

Road conditions, climatic conditions, driver attitude and country’s cultural habits are the factors that affect the travel time. In travel time studies, understanding the traffic factors that affects the travel time is essential for enchancing prediction accuracy. The SGQ performs skyline search for future journey schedule where travel time prediction is done by fuzzy inference engine by considering factors such as significance of date or day, journey time and climatic conditions on the day. The overall travel time in user specified spatial region by using a Fuzzy Inference Engine.

The work aims to provide travel time based advanced navigation solutions for user desired point of interest. The user can explore the location information based on constraints such as travel time limit, goal of journey, skyline objects. Nearest neighbors are retrieved by efficient novel algorithms and at the same time the user can choose best shortest path to reach the destination.

To sum up the following are the contributions:

1. Finds data objects for Skyline Geo-Calendar queries for future journey schedule by taking travel time as metric and computed based on day significance, climate conditions etc.
2. Travel time prediction on specified route is computed by using fuzzy inference rules and historical traffic data.
3. The overall travel time on the route for user specified future journey schedule and for desired POI is computed.
4. The cache technique which reduces the computation cost of future queries is introduced.

The reminder of this paper is organized as follows. In section 2, review of the related work on skyline queries and developing intelligent transportation system is presented. In section 3, Skyline Geo-Calendar query is defined in Road Networks. In section 4, explains the Fuzzy based approach for retrieving skyline data objects in road networks. Section 5 presents the results of the experimental simulation of the proposed algorithms with a variety of Spatial Network with large number of data and query objects. Finally section 6 focus on conclusion with future research scope.

## II. RELATED WORK

In the past, numerous algorithms Ugur Demiryurek et al (2000), Tao et al (2002), Papadias et al (2003), Samet et al (2008), Safar et al (2009) have gained importance to solve kNearest neighbour problem in Euclidean space. R-Tree structure is used by most of the algorithms for processing the queries. The moving object location updation is the main drawback of tree based structure. Later space based indexing structure (grid) are used. The distance between static objects is considered as a function of Euclidean distance.

In Papadias et al (2003) supported the kNN queries in spatial networks by introducing Incremental Network Expansion (INE) and Incremental Euclidean Restriction[IER] methods. To search minimum detour distance on the way to destination, In-Route Nearest Neighbor was first proposed by Shekar et al (2003) Voronoi diagrams to partition the spatial network to voronoi polygons, one for each data object was proposed by Kolahdouzanm et al (2004). Mi Young Jang et al (2012) propose a hybrid scheme to process an approximate k-Nearest Neighbor (k-NN) query by combining cloaking region based query processing. Through performance analysis, hybrid scheme outperforms the existing work in terms of both query processing time and accuracy of the result set.

The skyline operator was first introduced into the database community by Borzsonyi et al. [2]. Borzsonyi et al. [2] propose the Block-Nested-Loops algorithm (BNL) and the Extended Divided-and-Conquer algorithm (DC). Both algorithms processes the entire object set for retrieving the skyline data.

Skyline query processing has been studied extensively in recent years by Dellis et al (2007), Yiu et al (2005), Sharifzadeh et al (2006) Tao et al (2006). Papadias et al. [12] propose an R-tree based algorithm, Brand and Bound Skyline (BBS), which retrieves skyline points by browsing the R-tree based on the best-first strategy. BBS only visits the intermediate nodes not dominated by any determined skyline points. This method has more efficient memory consumption than the method in [6].

The privacy in spatial query is recently analysed by Donhee et al. (2015). The paper proposes a cloaking system model called anonymity of motion vectors (AMV) that provides anonymity for spatial queries. The proposed AMV minimizes the CR of a mobile user using motion vectors. AMV creates a ranged search area that includes the nearest neighbor (NN) objects to the queried who issued a CR-based query. The effectiveness of the proposed AMV is demonstrated in simulated experiments.

A Location Prediction Algorithm with Daily Routines in Location-Based Participatory Sensing Systems was proposed by Eric et al.(2014).This paper proposes a social-relationship-based mobile node location prediction algorithm using daily routines (SMLPR). The SMLPR algorithm models application scenarios based on geographic locations and extracts social relationships of mobile nodes from nodes' mobility. After considering the dynamism of users' behavior resulting from their daily routines, the SMLPR algorithm preliminarily predicts node's mobility based on the hidden Markov model in different daily periods of time and then amends the prediction results using location information of other nodes which have strong relationship with the node.

In Yong Wanga (2015), a new reverse skyline query processing method that efficiently processes a query over the moving objects is proposed. In addition, the proposed method processes a continuous reverse skyline query efficiently. In order to show the superiority of the proposed method, we compare it with the previous reverse skyline query processing method in various environments. As a result, the proposed method achieves better performance than the existing method.

Hyeong-Il Kim(2015) proposed a Hilbert curve-based cryptographic transformation scheme for spatial query processing on outsourced private data to improve privacy and reduce cost. A Hilbert curve-based cryptographic transformation scheme to preserve the privacy of the spatial data from various attacks on outsourced databases is designed. They also provide efficient range and k-NN query processing algorithms using a Hilbert-order index. The performance analysis confirms that the proposed scheme is robust against attack models and achieves better query processing performance than the existing cryptographic transformation scheme.

In this paper, a novel continuous query privacy-preserving framework in road networks. The framework is based on the concepts of k-anonymity and l-diversity. To achieve the quality of service, the distance limitation is taken into account. An Snet hierarchy based on the density of users, history traces, and road network topologies to accelerate the cloaking process performed at the anonymization server is designed. Two types of cloaking algorithms, for a single user and a batch of users, are designed. The security analysis shows that our framework is robust to typical attacks. The framework is evaluated from the aspects of privacy-preserving ability, quality of service, and system performance, which indicates that the system can provide good privacy protection while ensuring users׳ quality of service.

In summary, previous studies on skyline variants are limited to either Euclidean space or metric space for a specific location based application. In contrast, this work focuses on the travel time as a metric for skyline search which play a vital role in road networks. Travel time prediction is based on previous traffic data considering factors such as time slice of the day, day significance, climatic conditions etc by using fuzzy inference rules. It also helps future queries computation time by caching technique.

## III. SYSTEM MODEL

In this section, the road network and the skyline geo-calendar query search in spatial networks is defined. A spatial network [California Road Network], containing set of static data objects as well as query objects searching their skyline objects is considered as dataset. All road network data and daily traffic data are maintained by cloud server.

The underlying road network as a weighted directed graph G = (V,E) where E is an Edge set of road segments in the road network, V is the Vertex set of intersection points of the road segment and each edge is given travel time of its corresponding road segment as weights.

The Real Datasets for Spatial Databases of California network consists of Network's Nodes, Network’s Edges and Point of Interest (POI). The entire California road network extends from -114 to -124 degree longitudes and 32 to 42 degree latitudes.

The Network Node consists of Node ID, Longitude, and Latitude as shown in Table 3.1.

Table 1. California Road network nodes

| NODE ID | LONGITUDE | LATITUDE |
|---|---|---|
| 0 | -121.904167 | 41.974556 |
| 1 | -121.902153 | 41.974766 |
| 2 | -121.896790 | 41.988075 |
| 3 | -121.889603 | 41.998032 |
| 4 | -121.886681 | 42.008739 |
| 5 | -121.915062 | 41.970314 |
| 6 | -121.910088 | 41.973942 |
| 7 | -121.916199 | 41.969482 |
| 8 | -121.903198 | 41.968456 |

The sample nodes of Table 1 are shown as road network in the Figure 1. Node 0 lies at -121.904167 degree longitude and 41.974556 degree latitude position. Similarly, node 1 lies at -121.902153 degree longitude and 41.974766 degree latitude and node2 lies at -121.896790 degree longitude, 41.988075 degree latitude.

The Network Edge consists of Edge ID, Start Node ID, End Node ID, Distance and Travel time as shown in Table 2.The distance, travel time between the nodes are identified in the CRN and used as data. The edge 0 is a road that connects node 0 with node 1. Similarly, edge 1 connects node 0 and node 6. The length of edge 0 is 0.002025 kms and time taken to travel from node0 to node1 will be approximately 0.000506 seconds. Similarly, distance between node 0 and node 6 is the length of edge1 which is 0.005952 kms. The time taken to travel egde 1 is approximately 0.0001488 seconds.

Table 2. California Road network edges

| EDGE ID | START NODE ID | END NODE ID | DISTANCE (km) | TRAVELTIME(s) |
|---|---|---|---|---|
| 0 | 0 | 1 | 0.002025 | 0.0000506 |
| 1 | 0 | 6 | 0.005952 | 0.0001488 |
| 2 | 1 | 2 | 0.01435 | 0.0003587 |
| 3 | 2 | 3 | 0.012279 | 0.000307 |
| 4 | 3 | 4 | 0.011099 | 0.0002775 |
| 5 | 5 | 6 | 0.006157 | 0.0001539 |
| 6 | 5 | 7 | 0.001408 | 0.0000352 |

| | | | | |
|---|---|---|---|---|
| 7 | 5 | 8 | 0.012008 | 0.0003002 |
| 8 | 7 | 265 | 0.003213 | 0.0000803 |
| 9 | 8 | 298 | 0.005382 | 0.0001346 |
| 10 | 9 | 10 | 0.01294 | 0.0003235 |

The Points of Interest are the desired geo-locations which the user wishes to visit. The Table 3 shows the list of POI id with their corresponding category name.

Table 3. CRN Point of interest – categories

| POI CATEGORY ID | POI CATEGORY NAME |
|---|---|
| 0 | AIRPORT |
| 1 | ARCH |
| 2 | AREA |
| 3 | ARROYO |
| 4 | BAR |
| 5 | BASIN |
| 6 | BAY |

The Longitude and Latitude position of POIs are shown in Table 4. From the Table 4, the POI ARCH lies at -124.133331 degree longitude, 40.833328 degree latitude position. Similarly, the POI AIRPORT lies at - 115.906113 degree longitude, 33.492779 degree latitude.

Table 4. POI Category - longitude, latitude position

| LONGITUDE | LATITUDE | POICATEGORY ID | POI CATEGORY NAME |
|---|---|---|---|
| -124.133331 | 40.833328 | 1 | ARCH |
| -124.151939 | 41.065559 | 2 | AREA |
| -124.229721 | 41.835281 | 4 | BAR |
| -124.303062 | 40.651112 | 5 | BASIN |
| -115.832779 | 33.185558 | 6 | BAY |
| -115.843063 | 33.208611 | 3 | ARROYO |
| -115.906113 | 33.492779 | 0 | AIRPORT |

The POI lying on the edges is shown in the Table 5. The start node and end node of each POI is listed. The table also contains the attribute associated with each POI category. From the Table 5, the start node of POI ID 40 is 8760, end node is 8765 and the attribute of POI ID 40 is Red Wood Park. Similarly, the POI 30 which is of category Hospital lies at start node 8876 and end node 8882 with Resnic as attribute.

Table 5. CRN Point of interest location

| START NODE | END NODE | POI CATEGORY ID | ATTRIBUTE |
|---|---|---|---|
| 8760 | 8765 | 40 | Red Wood |
| 8760 | 8765 | 0 | Strip |
| 8876 | 8882 | 3 | Grande Valley |
| 8766 | 8767 | 5 | Saline |
| 8825 | 8865 | 4 | Q Inn |
| 8511 | 8512 | 30 | Resnic |

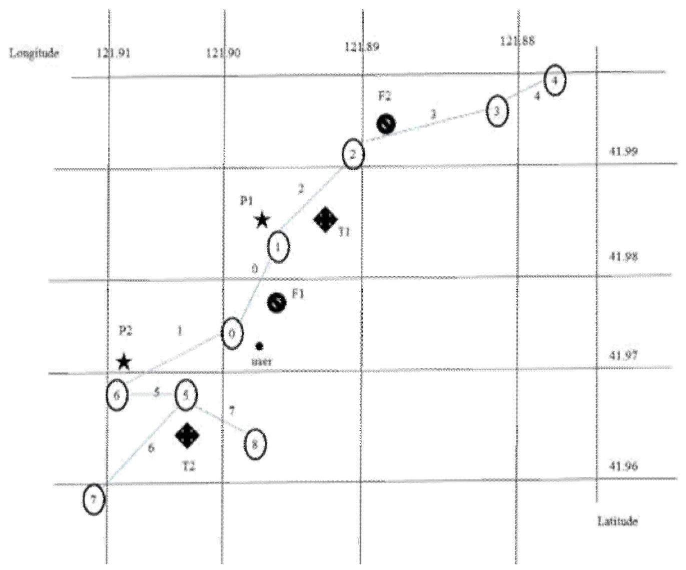


Figure 1. California Road Network

In Figure 1, the nodes 0,1,2,…etc are at corresponding longitude, latitudinal positions as per Table 1. The nodes are connected by edges and are given ids as 0,1,2,3...etc. P1, P2, F1, F2, T1, T2 are the POI on the road network. P1, P2 are pizza huts. F1, F2 are Fuel filling Stations. T1, T2 are Temples. The user is at 121.934215 and 41.9734789 latitude position. Suppose, if the user wishes to visit nearest Pizza Hut. The nearest road vertex of the user is taken as start point. As the node0 is the nearest road vertex of the user, the shortest path is calculated by taking node0 as start vertex. Then, based on the distance P1 is nearer to user as compared to P2. The final nearest neighbor for the user desired POI is P1 and path is node0 to node1.

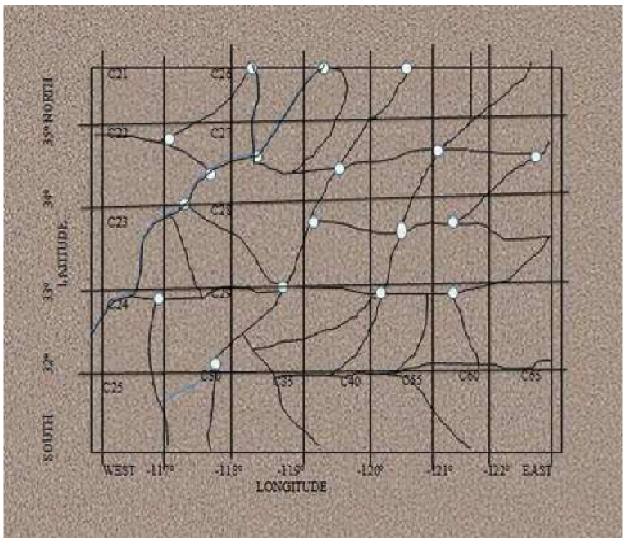


Figure 2. Nodes– clustering technique

The Figure 2 shows the sample of road network clusterization. The road network R is divided into C1, C2, C3,…., Cn cluster(s). The spatial region covered by one degree latitude, one degree longitude is considered as cluster size. The Table 6 shows the sample cluster classification of the road network.

The Node_Cluster_Map() functionfirst gets the total latitude and longitude count of California road network. Then, a loop is repeated which calculates the clusterID for each latitude and longitude pair. The corresponding clusterID is stored into Clustermaster table as shown in Table 6. The C1 cluster is mapped with (-116, 35) pair. Similarly, (-116, 34) is mapped to C2 cluster.

Table 6. Cluster classification

| Latitude | Longitude | ClusterID |
|---|---|---|
| 35 | -116 | C1 |
| 34 | -116 | C2 |
| 33 | -116 | C3 |
| 32 | -116 | C4 |
| 31 | -116 | C5 |
| 31 | -117 | C10 |
| 31 | -118 | C15 |
| 31 | -119 | C20 |
| 31 | -120 | C25 |
| 31 | -121 | C30 |

Each node in the nodemaster is updated with the corresponding clusterID for latitude and longitude value. Thus, the California road network is divided into clusters based on the spatial area covered by one degree latitude, one degree longitude size. The output of Node_Cluster_Map()function will be the mapping of each node with their corresponding ClusterID as shown in Table 7.

Table 7. Node classification

| Node_id | Longitude | Latitude | ClusterID |
|---|---|---|---|
| 20582 | -116.614166 | 35.751030 | C1 |
| 20583 | -116.616081 | 35.785458 | C1 |
| 20584 | -116.612167 | 35.797977 | C1 |
| 20622 | -116.617264 | 34.749596 | C2 |
| 20623 | -116.610237 | 34.748577 | C2 |
| 20643 | -117.203773 | 32.744373 | C9 |
| 20644 | -117.204697 | 32.742847 | C9 |

The node with ID 20582 with -116.614166 longitudes, 35.751030 latitude position belongs to Cluster C1. Also the node -116.616081, 35.785458 belongs to cluster C1 as the nodes have longitude degree -116 and latitude degree 35. This is given by the NetworkCluster_Map() classification algorithm 1.

**Algorithm 1 Function NetworkCluster_Map()**

```
//        function that clusters road network

1.  rc ->No. of latitude values
2.  cc -> No. of longitude values
3.  ch=1
4.  for each i < rc repeat thru step 7
5.  for each j < cc repeat thru step 7
6.  clusterID[ ]= “C”+ch
7.  clustermaster=(clusterID, latitude, longitude)
8.  ch -> ch+1
9.  for all nodes repeat thru step 11
10. nodeCID= UpdateCID( latitude,longitude)
11. stop
```

The road network clustering helps in optimizing the search time for every user query. If a user wishes to find his nearest hotel, the system helps him by retrieving k nearest point of interest and shortest path to reach each point of interest based on both road distance and travel time taken to reach it. To find desired point of interest, the initial nearest road vertex of the user is to be found. For this, the user Latitude, Longitude is mapped to a cluster Cj by this Node Cluster Mapping function.

Instead of searching entire database to find out minimum distance node from user location, the nodes in cluster Cj are matched for minimum distance node. The nodes within cluster Cj are sorted in descending order of their distances from user location (u) where the top most nodegives the nearest neighbour of the road vertex. This improves the efficiency of search and computation speed is increased. Harvesian Formula is used in finding the distance between two locations. Haversine Formula:

$a = \sin^2(\Delta\varphi/2) + \cos(\varphi 1).\cos(\varphi 2).\sin^2(\Delta\lambda/2)$
$c = 2.\text{atan2}(\sqrt{a}, \sqrt{(1-a)})$
distance = R.c
where $\varphi$ is latitude, $\lambda$ is longitude, R is earth's radius (mean radius = 6,371km)

**Query Definition**

A Skyline Geo-Calendar query is a skyline query on Time dependent road networks where data objects are returned in the relative to the user location and travel time is predicted from historical data. The SGQ is useful in following situations:

Example1:If a user wants to find a serviced apartment close to a hospital and a restaurant for a future journey schedule.

**Query Q1:**

$$\prod_{poi\_category = \text{"Apartment"}} (g_{min(traveltime)}\ \sigma (poi\_category = \text{"Hospital"}(poi\_master)) \cap g_{min(traveltime)}\ \sigma (poi\_category = \text{"Restaurant"}(poi\_master)))$$

Example2: If a tourist wants to find a low price restaurant nearer to a temple and close to a beach. For the query Q2, the algorithm returns restaurant (R1) object which is nearer to temple (T3) and close to beach (B1) based on the travel time.

**Query Q2:**

$$\prod(poi\_category = \text{"Restaurant"} \cup g_{min(price)})\ (g_{min(traveltime)}\ \sigma\ (poi\_category=\text{"Temple"}(poi\_master)) \cap g_{min(traveltime)}\sigma\ (poi\_category=\text{"Beach"}(poi\_master)))$$

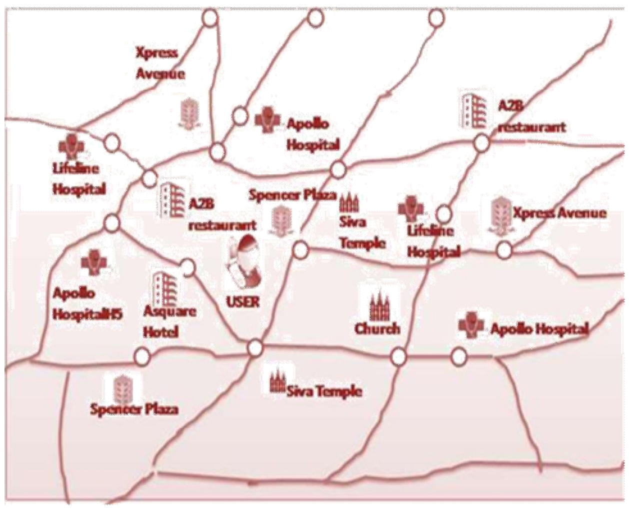


Figure 3. Spatial Road Network and Point of Interest

In the figure 3, R1,R2,R3 – restaurants, H1,H2,H3 – hospitals, B1,B2,B3 – beach).

For the query Q1, the algorithm returns serviced apartment (A3) object which is close to hospital (H4) and restaurant (R1) based on the travel time.

## IV. ARCHITECTURE

### 1. Phase I: Query Capture & Analyser

The SGQ architecture as shown in Figure 4, consists of four phases. In Phase I, the user query is captured and analysed. user current location nearest road vertex is computed. For this, the user location is given by GPS / Wi-Fi or any location specified by the user. The GetVertex() function returns the nearest neighboring road node "u" of the user location. Harvesian Formula is used in finding the distance between any two latitude and longitude distances. By applying the road clustering technique, the nearest vertex is retrieved by applying the user lat, long position values against function NodeMap() which gives the the nearest neighbor of "u".

### 2. Phase II: Query Tokenizer & Path Finder

During this phase II, the travel time for future journey date is predicted using fuzzy inference rules as shown in figure 5.

If traveldate is SATURDAY then
traffictype is SATURDAY(SAD)

If traveldate is SUNDAY then traffictype is
SUNDAY(SUD)

If traveldate is HOLIDAY then traffictype
is HOLIDAY(HOD)

If traveldate is SATURDAY and RAINY
then traffictype is
SATURDAY+RAINY(SAR)

The travel schedule includes both date of journey and time of journey. Traffic depends on various factors like day of travel, time, weather condition etc. Generally travel time depends on day significance, time of journey & climatic conditions etc. A Fuzzy Travel Time Forecaster inference engine in MATALB is designed and used for calculating the travel time. The Day significance is categorized as HOLIDAY, MONDAY, MID- WEEKDAY, FRIDAY, WEEKEND-SATURDAY, WEEKEND-SUNDAY etc. The journey time selection is based on following inference rules:

If traveltime is between 8hrs-11.30hrs
then traffictype is MORNING OFFICE HRS
(MOH)

If traveltime is between 11hrs- 13.30hrs
then traffictype is SHOPPING HRS (SPH)

If traveltime is between 13hrs-15.15hrs then
traffictype is NOON OFF PEAK HRS (OPH)

If traveltime is between 15hrs-17.10hrs
then traffictype is EVENING SCHOOL HRS
(ESH)

The function DateSignificance returns the datem apkey basing on the journey date significance(holiday or weekday or weekend etc). The function TraveltimeMap returns the slice of travel time from inference rules.

If traveldate is RAIN then
traffictype is RAINYDAY
If traveldate is CLOUDY then traffictype
is CLOUDYDAY

If traveldate is HEAVYRAIN then
traffictype is HEAVYRAINDAY

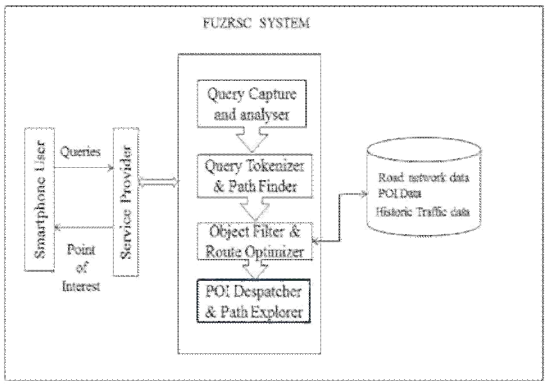

FUZRSC SYSTEM
Smartphone User
Queries
Point of Interest
Service Provider
Query Capture and analyser
Query Tokenizer & Path Finder
Object Filter & Route Optimizer
POI Despatcher & Path Explorer
Road network data
POI Data
Historic Traffic data

Figure 4. SGQ Architecture

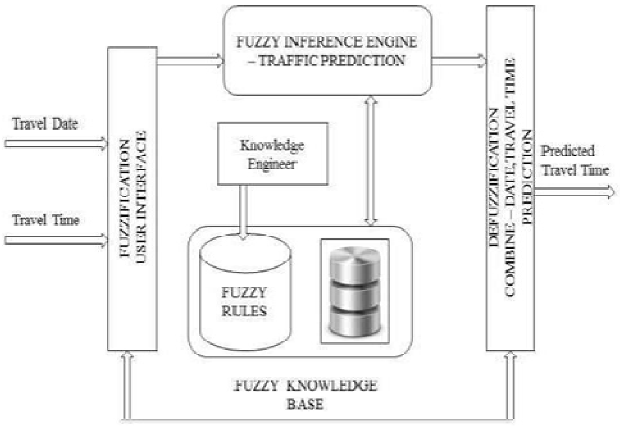

FUZZY INFERENCE ENGINE – TRAFFIC PREDICTION
Travel Date
Travel Time
FUZZIFICATION USER INTERFACE
Knowledge Engineer
FUZZY RULES
DEFUZZIFICATION COMBINE – DATE, TRAVEL TIME PREDICTION
Predicted Travel Time
FUZZY KNOWLEDGE BASE

Figure 5. Fuzzy Inference System

**Phase 4.3: PS Object Filter & Route Optimizer**

To find the user referred point of interest, all adjacent nodes from the nearest node are traversed. While searching the query points, the algorithm rejects the unwanted data points that are not within the boundary of the of the user preferences. From the first POI, the algorithm proceeds by considering other query preferences on the list of data objects. All the candidate objects are recomputed by considering the minimum travel time and climatic factors. At the end, the algorithm returns the goal directed skyline geo-calendar object relative to user location.

Optimization is achieved by computing the travel times to select POI which have the potential to participate in the final query result instead of all POI. The optimization would enhance the query processing by avoiding redundant computations and increase the efficiency of the algorithm. Aggregations for secondary query points are done before applying skyline retrieval. This helps in further reduction of query computation.

**Phase 4.4: POI Despatcher & Path Explorer**

The overall travel time is computed in the user specified spatial region. Each time a query is received, the cache stores the query point and path to reach the POIs. If the cache is exceeded in size, the least frequently used (LFU) query is evicted and new query result is stored in the cache. The cache is updated when there is a drastic change in travel time. The POI and shortest path to reach POI is despatched to the user in this phase.

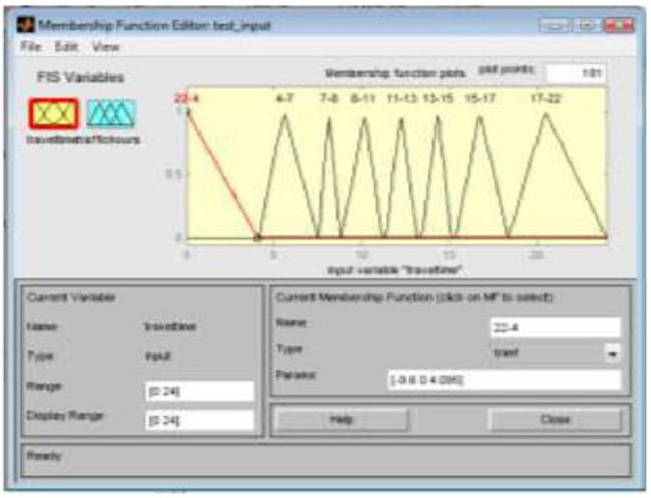
**Algorithm 2 : Skyline Geo-Calendar Query**

```
SGQ (uloc, Q, journeyschedule)
/* upos: query origin (latitude,ongitude) , Q – user query, POI[], an array consisting of user query preferences Q1,Q2,..Qn and K[], an array of their corresponding attributes a1,a2,..an, journey schedule: date and travel time of journey */

a.  Find the nearest vertex for the query origin uloc = Getvertex(uloc)
b.  R[i]={0} /*set the initial candidate objects as zero */
c.  POI[] = QueryCapture_analyser(Q)
d.  doj=datepredictor()
e.  ptime= traveltimepredictor()
f.  R[i]=Generate(Q) /* retrieves the list of initial candidate set for the query point Q1*/
g.  Path[i]= addQPtoheap(R[])
h.  if (checkresult(Path[]) ) then
i.  DispPOI_Path()
/* display skyline object */
j.  Cache(R[], Path[])

/* store for future queries*/
end if
```

## V. EXPERIMENTAL EVALUATION

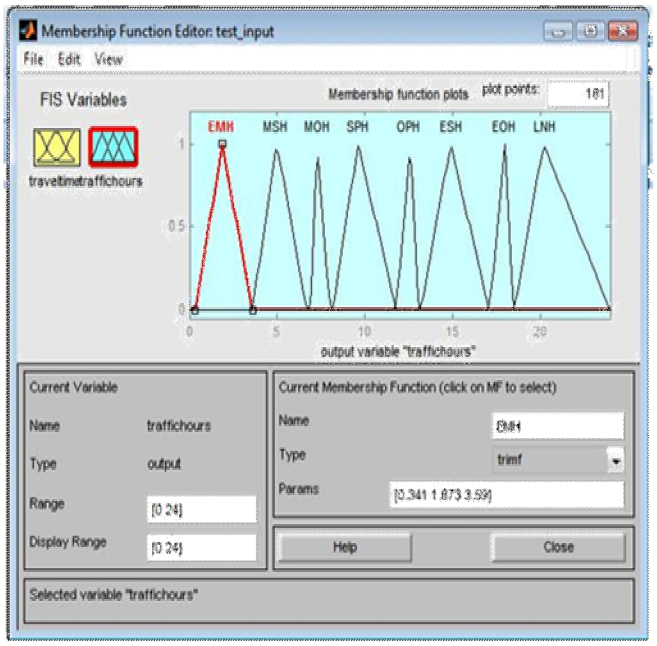
### I. Experimental Setup

The experimental setup are based on California road network which extends a 21,050 nodes and 21693 edges. The algorithm is implemented in Java and tested on Windows Platform with Intel Core 2 CPU and 80GB memory. The main metric adopted is CPU time that reflects how much time the algorithm takes in processing a query. The input map is taken from public databases available under Tiger/Line files. The experimental setup is based on an application scenario where a user at the starting location qs wishes to travel to a location qe on a future journey schedule. The travel time for future journey date is predicted using fuzzy inference rules using MATLAB as shown in Figure 6, Figure 7, Figure 8, Figure 9 and Figure 10. The travel schedule includes both date of journey and time of journey.



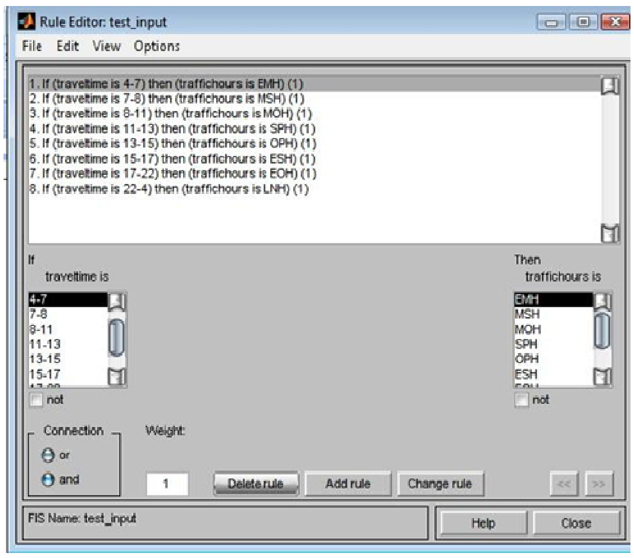
Current Variable
Name traveltime
Type input
Range [0 24]
Display Range [0 24]
Current Membership Function (click on MF to select)
Name 22-4
Type trapmf
Params [-0.8 0 0.4 0.095]
plot points: 181
Ready

Figure 6. Input membership function



Current Variable
Name traffichours
Type output
Range [0 24]
Display Range [0 24]
Current Membership Function (click on MF to select)
Name EMH
Type trimf
Params [0.341 1.873 3.59]
plot points: 181
Selected variable "traffichours"

Figure 7. Output membership function



```
1. If (traveltime is 4-7) then (traffichours is EMH) (1)
2. If (traveltime is 7-8) then (traffichours is MSH) (1)
3. If (traveltime is 8-11) then (traffichours is MOH) (1)
4. If (traveltime is 11-13) then (traffichours is SPH) (1)
5. If (traveltime is 13-15) then (traffichours is OPH) (1)
6. If (traveltime is 15-17) then (traffichours is ESH) (1)
7. If (traveltime is 17-22) then (traffichours is EOH) (1)
8. If (traveltime is 22-4) then (traffichours is LNH) (1)
```

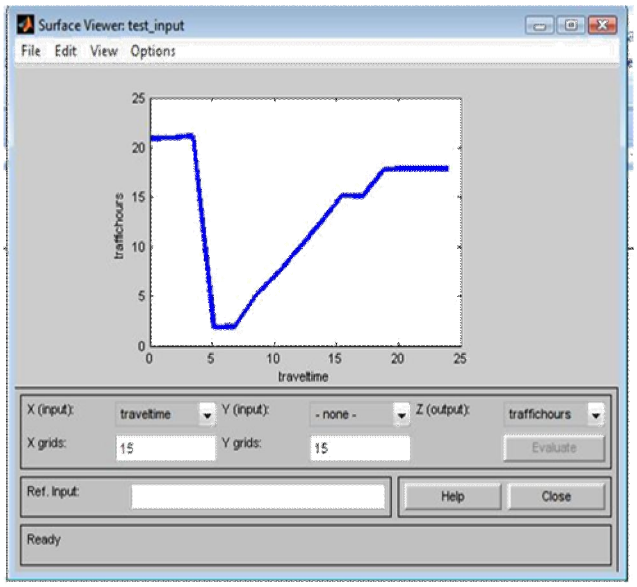
traveltime is: 4-7, 7-8, 8-11, 11-13, 13-15, 15-17

traffichours is: EMH, MSH, MOH, SPH, OPH, ESH

Weight: 1

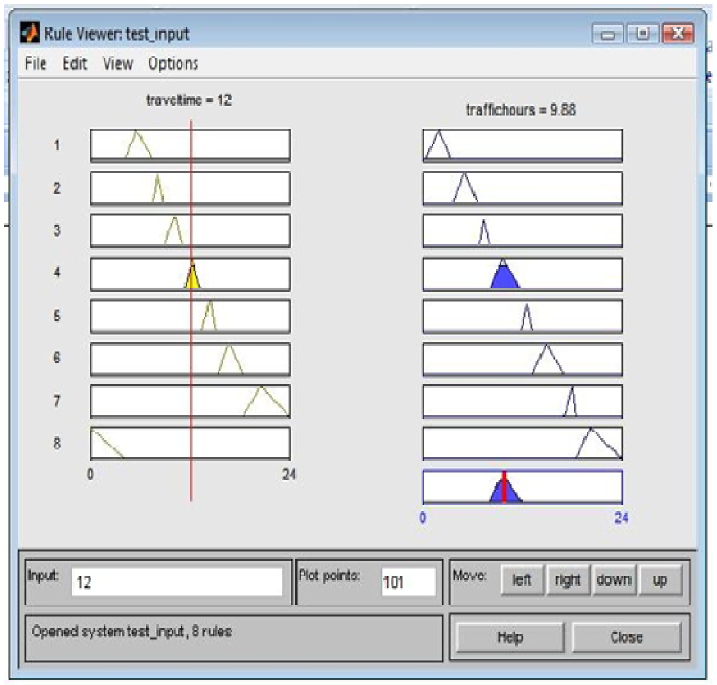
FIS Name: test_input

Figure 8. Fuzzy inference rules



X (input): traveltime
Y (input): - none -
Z (output): traffichours
X grids: 15
Y grids: 15
Ref. Input:
Ready

Figure 9. Rule viewer output



Input: 12
Plot points: 101
Opened system test_input, 8 rules

Figure 10. Surface viewer

## II. Performance Analysis

Query Q1: if a user wants to find a serviced apartment close to a hospital and a restaurant for a future journey schedule.

Travel time prediction for Query route Q1

The algorithm predicts travel time for a future journey schedule based on historic travel time. All the other algorithms use current travel time for execution.

Table 8. Travel time prediction for query route Q1

| Time slice | Original travel time | Predicted travel time |
|---|---|---|
| 8.00-8.15A.M. | 0.0007444 | 0.0007244 |
| 9.00-9.15A.M. | 0.00022471 | 0.000189471 |
| 11.00-11.15A.M. | 0.00081884 | 0.00065084 |
| 14.00-14.15P.M. | 0.001364733 | 0.001366733 |
| 17.00-17.15P.M. | 0.000909822 | 0.000809822 |
| 20.00-20.15P.M. | 0.000682367 | 0.000652367 |

Table 8 gives the travel time for the query Q1 on a particular day for all time slices. The predicted travel time is approximately 96% accurate.Between 8.00-8.15A.M., the travel time predicted is 0.0007244 for the user to reach the serviced apartment near to hospital and restaurant.

Figure 11 shows the travel time of original and predicted values for Query Q1. Using historical data for travel time prediction is helpful for tourists to know overall travel time depending on the traffic of the day.

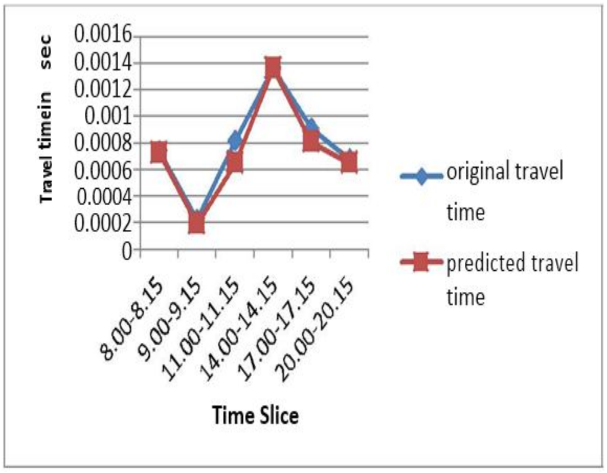


Figure 11. Travel time prediction for query route

Travel time prediction for Node n1

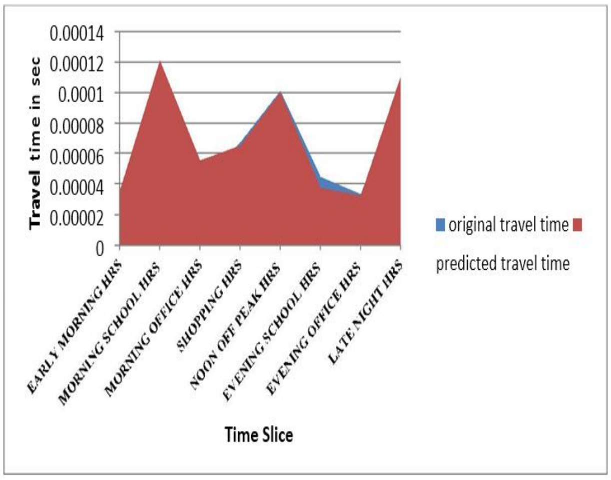


Figure 12. Travel time prediction for node n1

Figure 12 shows the travel time taken for any one node during the varying time slices like Shopping hrs, office hrs etc

Traffic Status for Query route Q1

Table 9. Traffic status for query route Q1

| Time Slice | Original Travel time | Predicted Travel time |
|---|---|---|
| EARLY MORNING HRS | NORMAL | LOW |
| MORNING SCHOOL HRS | LOW | PEAK |
| MORNING OFFICE HRS | PEAK | PEAK |
| SHOPPING HRS | PEAK | PEAK |
| NOON OFF PEAK HRS | LOW | WEEK |
| EVENING SCHOOL HRS | PEAK | PEAK |
| EVENING OFFICE HRS | PEAK | PEAK |
| LATE NIGHT HRS | LOW | WEEK |

Table 9 shows the predicted travel time in all time slices. It shows the predicted travel is almost same as original travel time.

Impact of cache in query computation

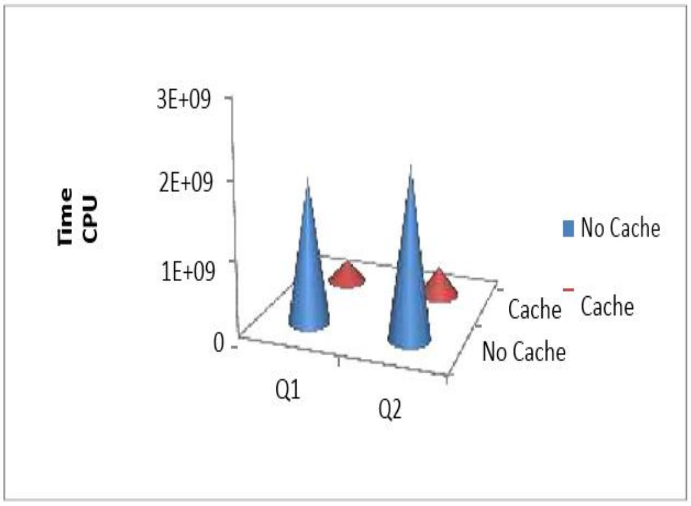


Figure 13. Impact of cache in query computation

Figure 13 shows the impact of Cache technique in computing future queries. For both queries Q1 and Q2, using of cache reduces the CPU time utilization.

Impact of query optimization on dominant objects

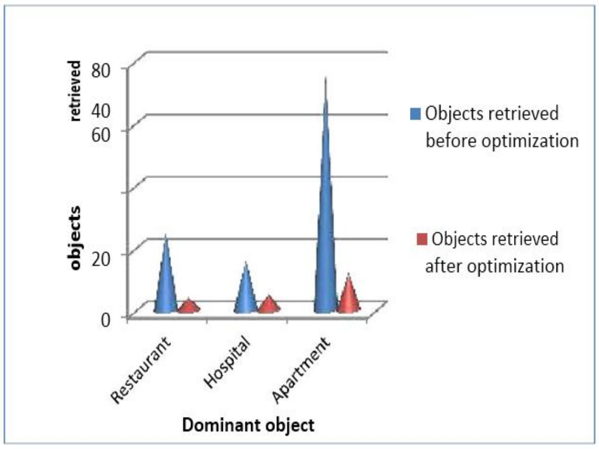


Figure 14. Impact of query optimization on dominant objects

Figure 14 shows the impact of query optimization in object retrieval from database. The POI namely Resturant, Hospital and Apartment retrieved before applying optimization and after optimization are shown in Figure 14.

## VI. CONCLUSION

In this paper, a Skyline Geo-calendar query on travel time dependent road networks is proposed. This algorithm efficiently searches and computes the data object to query origin. The travel time is predicted using historical traffic data based on fuzzy inference rules. Future work involves finding nearest neighbour for reverse skyline queries, range queries and aggregate queries.

**Conflict of Interests**

The authors declare that there is no conflict of interests regarding the publication of this paper.

## REFERENCES

[1] Borzsonyi S., Kossmann D. and Stoker K. (2001), “The Skyline Operator”, In: Proceedings of the 17th International Conferenceon Data Engineering, pp. 421-430.

[2] Dellis E. and Seeger B. (2007), “Efficient Computation Of Reverse Skyline Queries”, InProc. VLDB.

[3] K. Deng, X. Zhou, and H. T. Shen. 2007. Multi-source skyline query processing in road networks. In Proc. 23th Int. Conf.on Data Engineering, pages 796–805.

[4] Doohee Song, Jongwon Sim, Kwangjin Park, and Moonbae Song, “A Privacy-Preserving Continuous Location Monitoring System for Location-Based Services,” International Journal of Distributed Sensor Networks, vol. 2015, Article ID 815613, 10 pages, 2015. doi:10.1155/2015/815613

[5] Eric Ke Wang and Yunming Ye, “A New Privacy-Preserving Scheme for Continuous Query in Location-Based Social Networking Services,” International Journal of Distributed Sensor Networks, vol. 2014, Article ID 979201, 11 pages, 2014. doi:10.1155/2014/979201.

[6] Hyeong-Il Kim, Seungtae Hong, Jae-Woo Chang. Hilbert curve-based cryptographic transformation scheme for spatial query processing on outsourced private data. Journal of Network and Computer Applications. Volume 53, July 2015, Pages 57–73.

[7] Hui Zhu Su, En Tzu Wang and Arbee L. P. Chen. 2011. Continuous probabilistic skyline queries over uncertain data streams. Database and Expert Systems Applications, Springer.

[8] Khodayari A, Kazemi R, Ghaffari A,Braunstingl R. 2011. Design of an improved fuzzy logic based model for prediction of car following behavior. IEEE International Conference on Mechatronics (ICM), On Page(s): 200 – 205.

[9] Kolahdouzan M.R. and Shahabi C. (2004), “Continuous K-Nearest Neighbor Queries in Spatial Network Databases”, In STDBM.

[10] X. Lin, Y. Yuan, W. Wang, and H. Lu. 2005. Stabbing the sky: Efficient skyline computation over sliding windows. ICDE.

[11] D. Papadias, Y. Tao, G. Fu, and B. Seeger. 2005. Progressive skyline computation in database systems. ACM Trans.Database Syst., 30(1):41–82.

[12] J. Pei, B. Jiang, X. Lin, and Y. Yuan. 2007. Probabilistic skylines on uncertain data. In Proc. 33th Int. Conf. on Very LargeData Bases, pages 15–26.

[13] J. Pei, W. Jin, M. Ester, and Y. Tao. 2005. Catching the best views of skyline: a semantic approach based on decisive subspaces. In Proc. 31th Int. Conf. on Very Large Data Bases, pages 253–264.

[14] Samet H., Sankaranarayananj and Alborzih (2008), “Scalable Network Distance Browsing In Spatial Databases”, In SIGMOD.

[15] Sharifzadeh M. and Shahabic (2006), "The Spatial Skyline Queries", VLDB.

[16] Shekar S. and Yoo, J.S. (2003), "Processing In-Route Nearest Neighbor Search", In Proceedings Of GIS, pp. 9-16.

[17] F. Soriguera F. Robusté. 2011. Highway travel time accurate measurement and short-term prediction using multiple data sources. Transportmetrica, Volume 7, Issue 1, pages 85-109.

[18] Y. Tao, X. K. Xiao, and J. Pei. 2006. SUBSKY: Efficient computation of skylines in subspaces. In Proc. 22th Int. Conf. on Data Engineering, page 65.

[19] Tao Y. and Papadias D. (2002a), "Time-Parameterized Queries in Spatio-Temporal Databases", In SIGMOD.

[20] Ugur Demiryurek, Farnoush Banaei-Kashani and Cyrus Shahabi. (2010), "Towards K-Nearest Neighbour Search in Time-Dependent Spatial Network Databases", In: International Workshop on Databases in Networked Systems.

[21] M. Yang, Y. Liu, and Z. You. 2010. The reliability of travel time forecasting. IEEE Transactions on Intelligent Transportation Systems, vol. 11, no. 1, pp. 162–171.

[22] Yong Wanga, Yun Xiaa, Jie Houa, Shi-meng Gaoa, Xiao Niea, Qi Wangb. A fast privacy-preserving framework for continuous location-based queries in road networks. Journal of Network and Computer Applications. Volume 53, July 2015.

[23] Y. Yuan, X. Lin, Q., W. Wang, J. Xu Yu, and Q. Zhang. 2005. Efficient computation of the skyline cube. In Proc. 31th Int. Conf. on Very Large Data Bases.

[24] Tiger/Line: ww.census.gov/geo/www/tiger/.