Skyline Path Queries for Location-based Services

Nishu Chowdhury[1], Mohammad Shamsul Arefin[2,*]
Computer Science and Engineering Department, Chittagong University of Engineering and Technology
Chattogram, Bangladesh

***Abstract*—A skyline query finds objects that are not dominated by another object from a given set of objects. Skyline queries help us to filter unnecessary information efficiently and provide us clues for various decision making tasks. In this paper, we consider skyline queries for location-based services and proposed a framework that can efficiently compute all non-dominated paths in road networks. A path *p* is said to dominate another path *q* if *p* is not worse than *q* in any of the *k* dimensions and *p* is better than *q* in at least one of the *k* dimensions. Our proposed skyline framework considers several features related to road networks and return all non-dominated paths from the road networks. In our work, we compute skylines considering two different perspectives: business perspective and individual user's perspective. We have conducted several experiments to show the effectiveness of our method. From the experimental results, we can say that our system can perform efficient computation of skyline paths from road networks.**

***Keywords*—Skyline queries; trip planning; location-based services**

I. Introduction

Given a k-dimensional database DB, a skyline query retrieves a set of skyline objects, each of which is not dominated by another object. An object p is said to dominate another object p` if p is not worse than p` in any of the k dimensions and p is better than p` in at least one of the k dimensions. Fig. 1 shows a typical example of skyline. The table in Fig. 1 is a list of five routes, each of which contains two numerical attributes–"Cost" and "Distance". In the list, R2 and R5 are dominated by R3, while others are not dominated by any other routes. Therefore, the skyline of the list is {R1, R3, and R4}. Such skyline results are important for users to take effective decisions over complex data having many conflicting criteria. A number of efficient algorithms for computing skylines from the database have been demonstrated in the literature [1, 2, 3, 4, 5, 6].

Location-based services (LBSs) use positioning technology and traditional map information to furnish mobile users with new sorts of on-line services.

Location-based services in road network are becoming more popular. With rapid growth of technology, skyline queries on road networks [14, 15, 16, 17] have attracted much attention now a days.

Traffic jam refers to a long line of vehicles stuck in a jam. It is a common problem in the big cities and towns like Dhaka city of Bangladesh. Many factors such as less number of roads, lack of modern proper traffic management systems, narrowness of the roads, and increase of vehicles are the main causes of traffic jams in cities like Dhaka. These traffic jams are creating many problems such as not reaching in time at offices, ambulance carrying patients cannot reach at the hospitals in time etc.

In such a scenario, a well-developed location-based service that focuses on the road conditions such as traffic jam, number of passengers and cost to the destination can give some comforts to the people by choosing skyline routes from which people can select their desired paths based on their preferences.

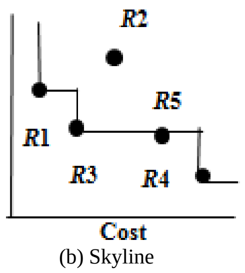
Each road in a road network has multiple-path criteria such as the distance of the road, the travel time through that road, the number of travellers and the number of traffic. The last two factors vary according to time. Before starting a journey, a traveller may want to know about the conditions of the road taken on their destination at a specific point of time. He/she may also want to know trip cost and other conditions of the roads.

In this paper, we apply skyline queries to support location-based services for road networks. In our approach, at first, a user needs to choose his pick-up point and the destination point. Based on the choice of the source and the destination by a user, our system then finds all alternate routes from source to destination. Next, each route is represented with several features such as traffic conditions, travelling time, travelling costs, number of passengers available through that each routes etc. After representing each route with a number of features, we apply skyline queries to filter dominated routes and to return only useful routes for the users. From the return results, a user can select his desired path such as less cost path or less traffic path.

The remainder of this paper is organized as follows: Section II provides a brief review of related work. We provide motivating examples at Section III. Section IV describes different concept related to the paper. We provide detail description of our proposed approach at Section V. In Section VI we present the experimental results. Finally, we conclude our paper at Section VII.

ID	Cost	Distance
*R*1	3	8
*R*2	5	4
*R*3	4	3
*R*4	9	2
*R*5	7	3

(a) Roads



(b) Skyline

Fig. 1. Skyline Example.

*Corresponding Author

II. Related Work

Since the introduction of skyline queries in 2001, there are many works related to skyline queries considering different settings.

The Block-Nested Loops Algorithm (BNL) [4], which is the easiest skyline query method. Its objective is to build a candidate skyline set. This calculation investigates every data point with each other data point in the dataset. The BNL calculation requires each data point in the database be checked and tried for predominance; consequently, the time required for calculation increments with the volume of information.

The DAC calculation [4] separates information into groups and at that point leads skyline query in each group. The results are consolidated to acquire a definitive result.

The SaLS aalgorithm [2] utilizes an element acquired from the raw information as a threshold value with which to filter and dispose data points.

The BBS algorithm [18] is at present the most well-known skyline query algorithm. The BNL and DAC algorithms require that a large portion of the data points be processed all together to complete the skyline query comparison. Conversely, BBS utilizes an index structure for the identification of skyline points, which diminishes the number of points that must be tested all together to process a query.

Previous studies in which skyline queries were utilized to check road networks can be classified into those attempting to recognize skyline landmarks and those trying to distinguish skyline paths.

Deng et al. [8] presented the idea of searching for skyline landmarks in street network. The skyline landmark query recognizes landmarks that coordinate user criteria when user is going on a road network. For instance, when a user travels on a road network, skyline landmark query encourages him/her those points that are adjacent. The algorithm in this work characterizes landmark attributes as static or dynamic. Static properties have fixed values. Dynamic properties have variable attributes. The algorithm initially distinguishes static skyline landmarks on their static attribute values. At that point, when users perform to check, the algorithm distinguishes all unique skyline landmarks dependent on their dynamic attribute values, and consolidates query points with the static skyline landmarks. At last, the algorithm can recover skyline landmarks that fit all characteristics.

Huang and Jensen [13] proposed an alternate skyline landmark search concept from that of [8]. They contended that users' movement in road networks ought to be founded on a recently settled way. The algorithm in this work was like that proposed by Deng et al. [8], which utilized the ideas of static and dynamic attributes to identify skyline landmarks. The main contrast between the algorithms is the attribute calculation method. Deng et al. [8] considered the separation between the landmark and the inquiry area of the user, while the researchers of this work consider the separation between the landmark and the path preset by the user.

Tian et al. [21] presented the idea of skyline paths. Their proposed algorithm would utilize the edge attributes of a road network to discover all skyline paths between the user-specified starting vertex and goal. The algorithm would first decide a single skyline path between a starting vertex and goal whose summation of all attributes values is the most reduced among all ways. At that point, the algorithm would recognize other skyline paths by (1) a greedy algorithm to locate a relay vertex between starting vertex *s* and goal *t*. If skyline path domination was available after adding the two values, at that point *a* can't be a piece of a skyline path. In this instance, the algorithm again employs the greedy algorithm to identify other possible relay vertices or identify the next relay vertex following *a*.

Kriegel et al. [15] utilized the greedy algorithm to distinguish a possible relay vertex among *s* and *t*. Kriegel et al. [15] utilized a reference vertex to help estimations. By utilizing such a technique, they proclaimed the strategy proposed in this work was quicker than that proposed in crafted by [21].

Many researchers have looked to broaden the works in [15] and [21]. Aljubayrin et al. [1] examined the issue of skyline trips on different POI classes. Hsu et al. [10] connected the possibility of a skyline path to the arranging of treks to beat the conventional problem of acquiring multicriteria answers. Yang et al. [23] consolidated GPS history information in their inquiries to enable the user to design their skyline route under time-varying vulnerability. Unfortunately, these works don't consider aggregate attributes in road networks, which make them inapplicable to the issues tended to in this examination.

A new concept M-tree structure is described in [7]. A. Guttman al. [9] describes dynamic index structure called an R-tree and W. Son al. [20] describes spatial skyline queries for dynamic environment.

In [11], they focus on processing the continuous skyline query in road networks. They design a grid index to effectively manage the information of data objects. They proposed several algorithms combined with the grid index to answer the skyline queries.

In [12], they overcome the specific assumptions that each object is static in road networks. They focus on processing the CKNSQ over moving objects with uncertain dimensional values in Euclidean space and the velocity of each object (including the query object) varies within a known range.

Sheng et al. [19] present external memory algorithms for solving the skyline problem its variants in a worst-case efficient manner. They proved that the running time can be improved if some dimensions have small domains.

In [22], they bring out novel information by analyzing bulky databases to consolidate users experience to find place of interest. They use Apriori algorithm for identifying hidden association among item sets from large databases of user checking in data and to construct the route analogous to the key terms provided by user.

III. Motivating Examples

Consider the graph of Fig. 2 that represents a road network where *L1* is considered as a source location and *L2* is a destination location. Each vertex in Fig. 2 represents a junction i.e. dropping and/or pickup point and each edge represents a connection between two vertices. There are four values associated with each edge those are travel time, cost, distance and passengers, respectively. For example, edge (*L1, l1*) has values (3.18, 0.35, 0.7, 4), which indicates that the required time to reach from L1 to l1 is 3.18, cost of is 0.35, the distance between *L1* and *l1* is 0.7 and number of available passengers in is 4. In Fig. 2 if we use the route <*L1, l5, l6, l2, l8,l9,l4, L2*> to reach from *L1* to *L2*, we need total time 3.18 + 5 + 2.27 + 3.18 + 4.54 +4.09 + 1.81 = 24.07 and cost is 0.35 + 0.55 + 0.25 + 0.35 + 0.5 + 0.45 + 0.2 = 2.65. Here, total distance is 0.7 + 1.1 + 0.5 + 0.7 + 1 + 0.9+0.4 = 5.3 and the number of available passengers in this route is 4 + 5+ 7 + 6 + 8 + 10 + 10 = 50.

In this paper, we have considered two different scenarios. One is for business purpose and another is for individual user's perspective. These scenarios are explained below.

A. *Business Perspective*

In applications such as Pathao and Uber, one trip can only be allotted to one passenger request at a specific time. In contrast, microbuses and cars have the capacity to carry five to eight passengers, respectively. Let us assume that a service provider has a microbus, which is a 10-seater vehicle. Suppose this person plans a trip from location *L1* to *L2*. Before starting the journey, by utilizing our method this person can find shorter route as well as a faster route. Our method also suggests a route having a large number of passengers, whereas for a fast route this method provides a route with a shorter distance and lesser traffic. Here, the passengers are those whose destination location is the same as that of the service provider and the start location belongs to the list of suggested routes. Thereby, the service provider can choose its preferable route and accept the passenger request for the same.

Table I shows the distance information on all routes and Fig. 3 presents the traffic and passenger conditions of two alternate routes. In Fig. 3, the graph lines are represented by three colours: green represents light traffic, orange represents medium traffic, and red indicates high traffic.

Hence, compared to multiple routes, it is necessary to find a desired route that is not dominated by any other route. In detail, a route is preferable to visitors if it is not dominated by any other route. The information on routes is given below. This information is collected from Google Map API.

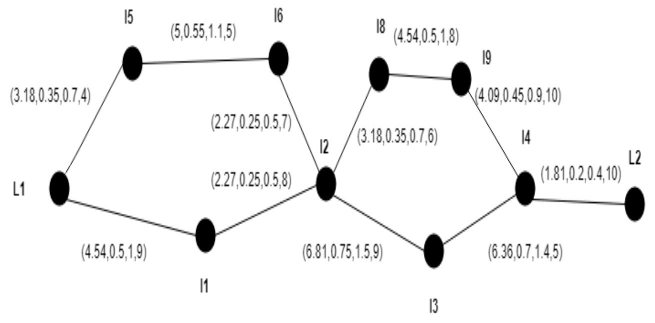


Fig. 2. Example of a Graph Representing a Road Network.

TABLE I. INFORMATION ON ROUTES

Route	Starting	Ending	Distance	Locations
Route 1	*L1*	*L2*	5.3 Km	*L1,l5,l6,l2,l8,l9,l4,L2*
Route 2	*L1*	*L2*	4.8 Km	*L1,l1,l2,l3,l4,L2*
Route 3	*L1*	*L2*	5.6 Km	*L1,l5,l6,l2,l3,l4,L2*
Route 4	*L1*	*L2*	4.5 Km	*L1,l1,l2,l8,l9,l4,L2*

The traffic and passenger conditions of two routes are graphically represented in Fig. 3. The route line colour changes with time. From this figure, we can say that there is a light traffic for Road 1 from 8.00 a.m. to 12 p.m., and Road 2 will be free after 2 p.m. It is also observed that there is heavy traffic for Road 1 from 3 p.m. and for Road 2 from 11 a.m. to 2 p.m. This graph is also helpful in tracing a medium traffic condition. Road 1 has medium traffic from 12 p.m. to 3 p.m. and Road 2 from 8 a.m. to 11 a.m. We can also calculate our travel cost from route distance.

Table II represents the traffic and passenger conditions of every location for Route 1 and 2. Each row in the table represents traffic and passenger conditions. For measuring passenger we use normalize value. These are helpful in identifying the most interesting and preferable route.

Another graphical representation is given below. Fig. 4, represents the passenger condition with traffic. In this figure, blue colour Route 1 and green Road 2.

From Table II we find skyline points H3, H4.

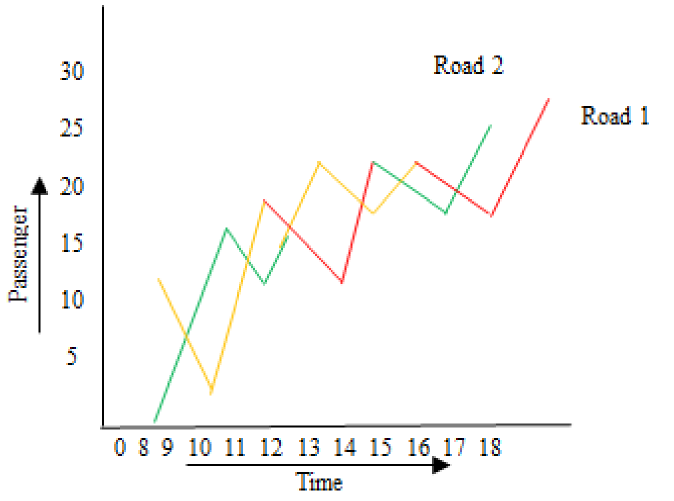


Fig. 3. Traffic Condition of Two Routes with Respect to Time.

TABLE II. ROAD CONDITIONS FOR DIFFERENT TIME INTERVALS

ID	Route	Travel Time	Passenger	Traffic	Distance (Km)
H1	Route 1	8.00 am	50	Low	5.3
H2	Route 1	2.00 pm	40	High	5.3
H3	Route 2	8.00 am	41	High	4.8
H4	Route 2	9.00 pm	35	Medium	4.8

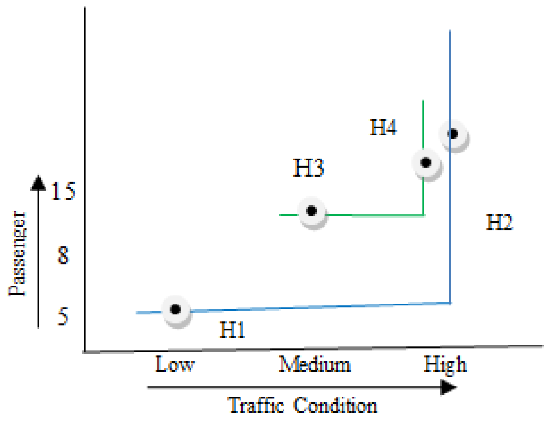


Fig. 4. Traffic and Passenger Conditions.

B. Individual Perspective

Suppose a visitor wishes to travel from Location *L1* to *L2*. Before starting their journey, he/she wishes to know about the route and traffic conditions as well as the cost of travel. Then, this method will provide him/her with traffic information using Google Map Traffic API and calculate the cost by calculating the fuel cost per litre, the mileage of their vehicle and the distance between their start and end locations. From the resulted dataset a user can easily filter routes according to choice.

Table III represents road condition for specific interval. Each row in the table represents traffic and cost, which are helpful to identify the most interesting and preferable route.

From Table III, we find that cost is changing with user. Moreover, cost depends on user vehicle's fuel cost and mileage.

In this paper, we compute a method that can help service providers to choose their desired route from our resulted skyline routes. Our location-based computation method can significantly find the appropriate route, based on the dataset. By this way, our method is useful for individual trip planning and transport service business planning.

TABLE III. TRAVEL COST

ID	User	Start point	End point	Route	Distance (Km)	Traffic	Cost
H1	user 1	*L1*	*L2*	Route 1	5.3	High	2.65
H2	user 1	*L1*	*L2*	Route 2	4.8	Low	2.4
H3	user 1	*L1*	*L2*	Route 3	5.6	High	2.5
H4	user 1	*L1*	*L2*	Route 4	4.5	High	2.25
H5	user 2	*L1*	*L2*	Route 1	5.3	High	3.65
H6	user 2	*L1*	*L2*	Route 2	4.8	High	3.4
H7	user 2	*L1*	*L2*	Route 3	5.6	Low	3.5
H8	user 2	*L1*	*L2*	Route 4	4.5	Low	3.35

IV. PRELIMINARIES

Consider a database *DB* with *N* attributes and *k* objects. Let *a1, a2,...,aN* be the *N* attributes of *DB*. We consider that smaller values in each attribute are better and that each attribute has positive values.

A. Skyline Queries

Skyline query is a decision-supporting mechanism that highlights the best options among vast data.

An example is given below:

In Fig. 5, we have some points in a two-dimensional space, as shown above, then we define a point *p* that will dominate point *q* provided its coordinates are larger than that of *q*. In this example, there is a point *p* that dominates several other points. So what is the skyline point? Skyline points are points that are not dominated by any other points present in the dataset. They are also called maximal points. If you connect these with horizontal and vertical lines, then you will get skyline points.

Let *L* denote a set of all locations. Each location has an ID and a spatial coordinate *l = (xy)*. Let us suppose *A* is a category attribute. In our research work, passenger and traffic are category attributes. So, we denote the coordinates of location *L* by *l. L*; *l.a* represents the value of attribute *A.*

Definition 1 (Dominance Relationship): Given two objects *a* and *a'* exist, then object *a* is said to dominate *a'* if *a < a'* for all the attributes.

Definition 2 (Skyline Query): Skyline query is the set of objects that cannot be dominated by any other object. Given point *p, r* ∈ *D*. If *p < r*, then *p* belongs to the skyline set.

B. Multi-Attribute Network Graph (MAG)

Graph *G (V, E, W)* is a multi-attribute network graph, where *V* denotes a set of vertices, *E* a set of edges, and *W* weight vector. In Fig. 6, nodes define profiles of activity, roles and actors etc. Edges define the relationship among those nodes or entities and weight defines the behaviour of the edges.

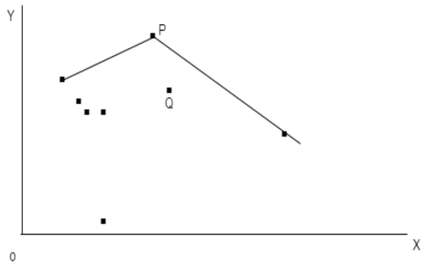


Fig. 5. Skyline Points.

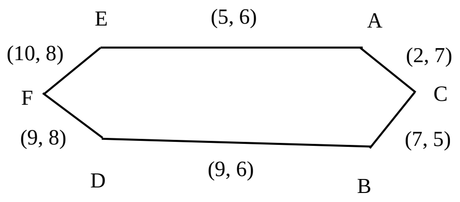


Fig. 6. Multi-Attribute Network Graph (MAG).

V. Methodology

Our method comprises two modules: the first module delves into the business perspective and the second into the individual perspective. Fig. 7 describes the proposed framework. In each module, the user can provide the source and destination addresses while prioritising a specific destination based on his/her choice.

The business perspective and individual perspective operates in three processes or modules: processing module, query execution module and output module. Our functional algorithm, which parses the dataset and the filters, is known as the processing module. In terms of both perspectives, it works in five steps: first, it measures the geolocations of the start and end locations. Upon completion, an iteration process continues to measure all alternate routes from the source location to the destination location. Thereafter, it calculates traffic, trip cost or passengers based on the dataset. Thereafter, the process migrates into the query execution module, where a resulted dataset is generated imposing skyline queries. Through these processes, we get the dominant paths that are filtered later on the system output, which shows the result of these potential paths.

The most naïve approach to locating skyline paths in a road network is to identify all of the paths between the origin and destination in the network, calculate attributes of the paths, and perform a dominance check of all the attributes. The process of estimating traffic, cost and passenger are given in below.

A. *Traffic Estimation*

Fig. 8 describes the traffic condition at a specific time. Suppose we wish to assess the traffic conditions in all alternate routes from *L1* to *L2* at 2 pm. In this framework, car has been used as a transport mode. We get four routes from the given graph: Route 1 comprises *L1, l5, l6, l2, l8, l9, l4, L2* and Route 2 comprises *L1, l1, l3, l4, L2*. Similarly Route 3 contains *L1, l5, l6, l2, l3, l4, L2* and Route 4 contains *L1, l1, l2, l8, l9,l4,L2*. For assessing the traffic conditions at a specific time, Google Map Traffic API is used. For example, if someone wants to assess the traffic condition from *L1* to *L2* at 2 pm, then the system counts all alternate routes that he/she can take to reach the destination. Thereafter, it uses the latitude and longitude of a distance at every 0.5 km interval and checks the location at each iteration. Whenever a new location returns, we measure the traffic condition at those points by employing Google Map Traffic API. It provides the standard time and the time required to reach one's destination, and then it stores all the data on the latter for every 0.5 km interval. In this way, we can obtain all the data on the time taken for all alternate routes. In this figure, the blue-coloured text represents the standard time (in minutes) taken to travel from one location to another. Another colour represents the time required to travel from one location to another. In this figure, three different colours are used: orange is used to represent medium traffic, green to indicate low traffic and red for heavy traffic. When the standard time is equal to the required time, the given time interval contains medium traffic. If the required time is low, it indicates the presence low traffic. Otherwise, the presence of heavy traffic is indicated. By this way, we gather data for our dataset. Tables IV, V, VI and VII represent the required time for Route 1, Route 2, Route 3 and Route 4 respectively.

Now, we calculate the total required time for each route. From above dataset we find Route 1, Route 2, Route 3 and Route 4 require 24.07, 21.79, 25.43 and 20.43 minutes respectively.

B. *Cost Estimation*

Table III represents user wise cost for each route. Here, we represent how cost is changing with distance, mileage and fuel consumption. In our method, cost measures by using the following formula.

$$\text{Cost} = (\text{mileage/per ltr fuel cost}) * \text{Distance} \tag{1}$$

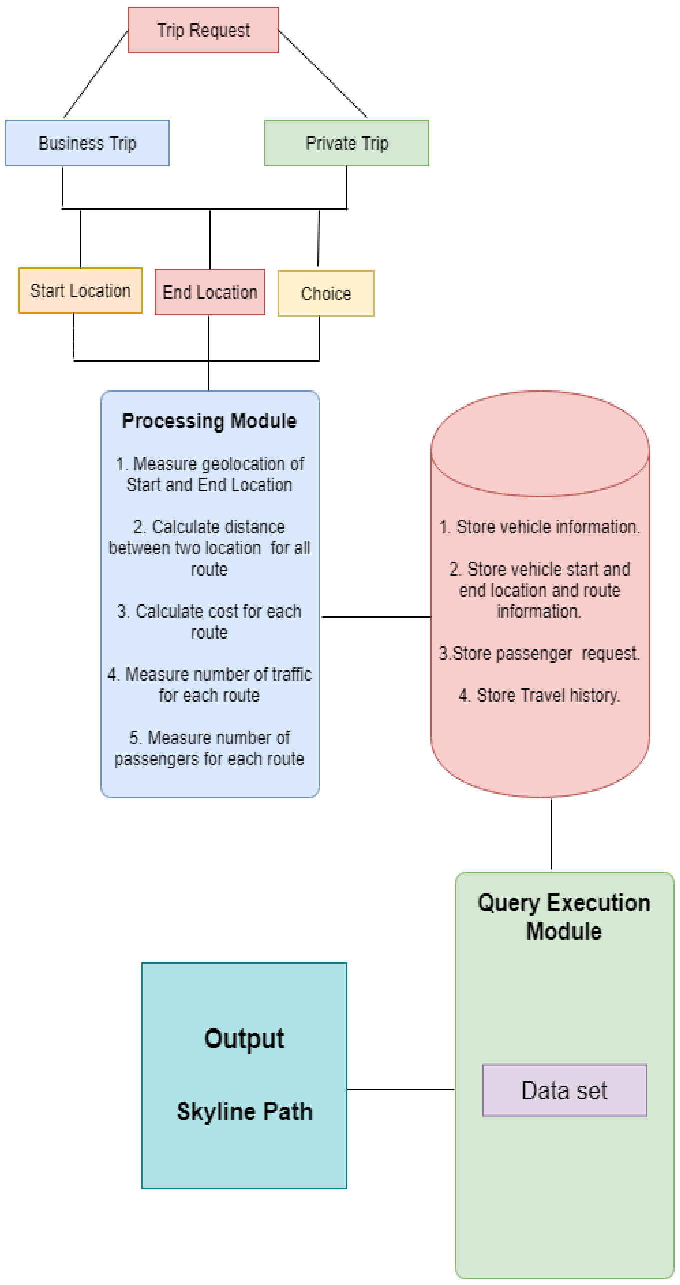

Trip Request
Business Trip
Private Trip
Start Location
End Location
Choice
Processing Module
1. Measure geolocation of Start and End Location
2. Calculate distance between two location for all route
3. Calculate cost for each route
4. Measure number of traffic for each route
5. Measure number of passengers for each route
1. Store vehicle information.
2. Store vehicle start and end location and route information.
3.Store passenger request.
4. Store Travel history.
Query Execution Module
Data set
Output
Skyline Path

Fig. 7. System Architecture.

TABLE IV. REQUIRED TIME AND DISTANCE FOR ROUTE 1

Source	Destination	Required Time (min)	Distance (Km)
L1	l5	3.18	0.7
l5	l6	5	1.1
l6	l2	2.27	0.5
l2	l8	3.18	0.7
l8	l9	4.54	1
l9	l4	4.09	0.9
l4	L2	1.81	0.4

TABLE V. REQUIRED TIME AND DISTANCE FOR ROUTE 2

Source	Destination	Required Time (min)	Distance (Km)
L1	l1	4.54	1
l1	l2	2.27	0.5
l2	l3	6.81	1.5
l3	l4	6.36	1.4
l4	L2	1.81	0.4

TABLE VI. REQUIRED TIME AND DISTANCE FOR ROUTE 3

Source	Destination	Required Time (min)	Distance (Km)
L1	l5	3.18	0.7
l5	l6	5	1.1
l6	l2	2.27	0.5
l2	l3	6.81	1.5
l3	l4	6.36	1.4
l4	L2	1.81	0.4

TABLE VII. REQUIRED TIME AND DISTANCE FOR ROUTE 4

Source	Destination	Required Time (min)	Distance (Km)
L1	l1	4.54	1
l1	l2	2.27	0.5
l2	l8	3.18	0.7
l8	l9	4.54	1
l9	l4	4.09	0.9
l4	l2	1.81	0.4

C. Passenger Estimation

Fig. 9 describes the condition of passenger at specific time; suppose, we need to assess the condition of passenger of all alternate routes from location *L1* to *L2*. The passenger condition for specific time for each location can be assessed through passenger request. Fig. 8 shows the passenger condition. From this Fig. 8, we have found four alternate routes: Route 1 comprises L1, l5, l6, l2, l8, l9, l4, L2 and Route 2 comprises L1, l1, l3, l4, L2. Similarly Route 3 contains L1, l5, l6, l2, l3, l4, L2 and Route 4 contains L1, l1, l2, l8, l9,l4,L2. For example, we want to measure number of passengers for Route 1.At first, we count passengers of all location of Route 1, whose destination location is *L2*. Now get the maximum value from these locations. Suppose the value is *P*. We use the following formula to normalize location wise passengers.

$$P_i = P + 1 - P_i \tag{2}$$

Tables VIII, IX, X and XI represent the condition of passengers for Route 1, Route 2, Route 3 and Route 4, respectively.

Now, calculate the condition of passenger for each route. Route 1, Route 2, Route 3 and Route 4 has 50, 41, 40 and 51 passengers, respectively.

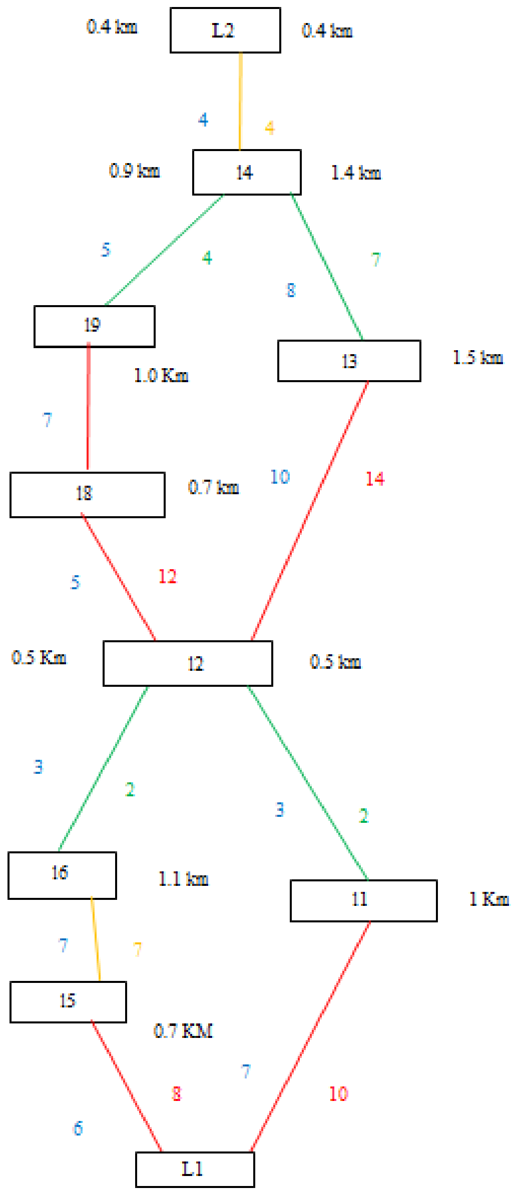


Fig. 8. Traffic Condition and the Distance of all Alternate Routes from *L1* to *L2*.

TABLE VIII. PASSENGER CONDITION FOR ROUTE 1

Source	Destination	Number of Passengers
L1	*l5*	4
l5	*l6*	5
l6	*l2*	7
l2	*l8*	6
l8	*l9*	8
l9	*l4*	10
l4	*L2*	10

TABLE IX. PASSENGER CONDITION FOR ROUTE 2

Source	Destination	Number of Passengers
L1	*l1*	9
l1	*l2*	8
l2	*l3*	9
l3	*l4*	5
l4	*L2*	10

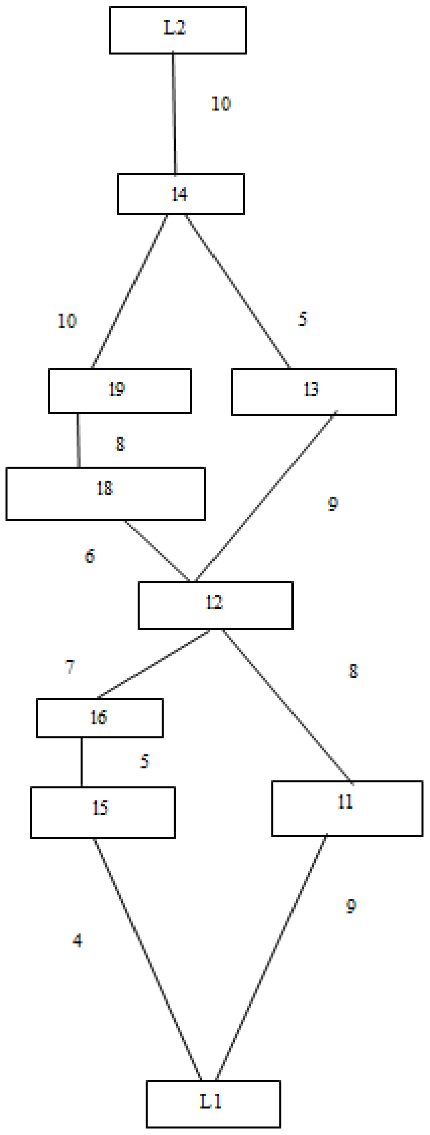


Fig. 9. Passenger Condition from L1 to L2.

TABLE X. PASSENGER CONDITION FOR ROUTE 1

Source	Destination	Number of Passengers
L1	*l5*	4
l5	*l6*	5
l6	*l2*	7
l2	*l3*	9
l3	*l4*	5
l4	*L2*	10

TABLE XI. PASSENGER CONDITION FOR ROUTE 2

Source	Destination	Number of Passengers
L1	*l1*	9
l1	*l2*	8
l2	*l8*	6
l8	*l9*	8
l9	*l4*	10
l4	*L2*	10

D. Computing Skyline

Here, we generate a dataset that measure attributes such as traffic, passenger, cost and distance.

Let's consider a scenario. Suppose source is *S* and destination is *D*. There are ten alternate routes from *S* to *D*. We denote traffic condition as low, medium and high and define them as 1, 2, and 3 respectively. Table XII represents the route condition for a specific time.

From this dataset we need desire routes. By using BBS [18] algorithm we get our skyline routes. Fig 10 describes the BBS algorithm.

Using BBS algorithm, we get our skyline routes as *R1*, *R2*, *R3* and *R4*. From this method, a user can easily find his/her desire route in proficient and appropriate way. If one wants a large passenger, low traffic and low cost route, then he/she can get the desired routes from the resulted routes.

TABLE XII. ROAD CONDITION FOR A SPECIFIC TIME

Route	Traffic	Cost	Passenger	Distance (Km)
R1	1	55	20	20
R2	3	60	9	10
R3	2	50	10	15
R4	1	45	8	30
R5	3	100	20	50
R6	2	120	30	60
R7	2	110	50	70
R8	2	130	40	50
R10	2	150	50	30
R11	3	140	60	40

Algorithm 1: BBS
Input: A dataset *D* (r-tree).
Output: The Set of skyline points of dataset *D*.

```
1. S=Ø // list of skyline points
2. insert all entries of the root R in the heap
3. while heap not empty
4.   remove top entry e
5.   if e is dominated by some point in S discard e
6.   else // e is not dominated
7.     if e is an intermediate entry
8.       for each child e_i of e
9.         if e_i is not dominated by some point in S
10.          insert e_i into heap
11.    else // e is a data point
12.      insert e_i into S
13. end while
14. end
```

Fig. 10. BBS Algorithm for Skyline Computation (Adapted from [18]).

VI. Experiments

We have implemented our proposed system in .Net Framework. We have performed the experiment in a simulation environment of a PC running on windows OS having an Intel(R) Core i7, 1.73 GHz CPU and 4 GB main memory. Due to the lack of real data, we evaluate our proposed algorithm using synthetic datasets only.

Fig. 11 shows the results when we consider Route 1, Route 2, Route 3 and Route 4. We observe that with the increases of distance, number of passengers varies.

Fig. 12 shows that with the increase of distance, number of routes also increases.

In Fig. 13, when we consider two (2D), three (3D), four (4D), and five (5D) features. We observe that with the increases of routes, there is very slight increase in computation time. This is because during the computation process, time increases with the increase of number of routes. We can also observe that computation time gradually increases if the number of features increases.

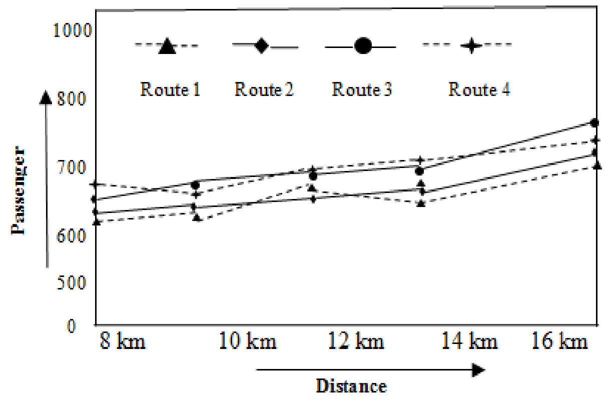


Fig. 11. Passenger Varies with Distance.

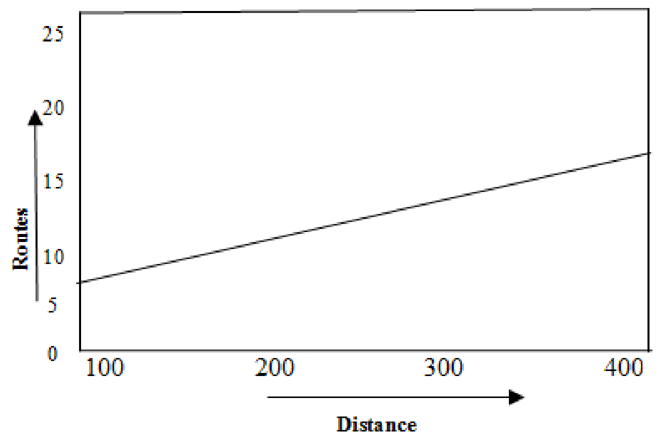


Fig. 12. Number of Routes Varies with the Distance.

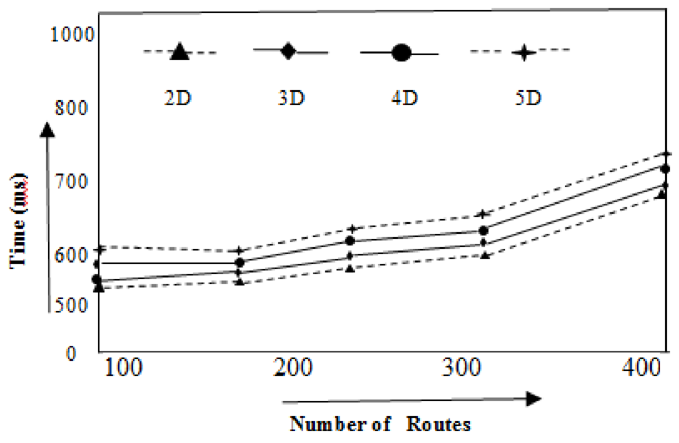


Fig. 13. Time Varies with Number of Routes.

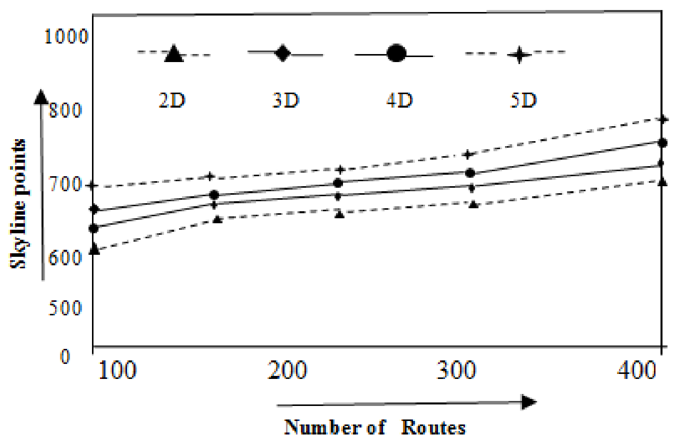


Fig. 14. Skyline Points Varies with Number of Routes.

Simultaneously, it is also observed that skyline points increase with the number of routes and number of features. Fig. 14 represents how skyline points increase with the number of routes.

VII. CONCLUSION

With the rapid growth of civilization, traffic is seen to increase day by day. Therefore, collecting traffic information, passenger condition and cost calculation has become a popular method. Our experimental results demonstrate that the proposed algorithm is scalable enough to compute the skyline path for a specific time. The proposed approach can easily expand for recommendation. In this work, we performed different analyses on synthetic data. In future, we aim to expand large passenger route methodology in more efficient way and find desire route based on user preference. So that we can get skyline points in more proper ways. We also want to trace the vehicle movement and position in a more efficient and effective way.

REFERENCES

[1] S. Aljubayrin, Z. He, and R. Zhang, ‘‘Skyline trips of multiple POIs categories’’, in Proc. Int. Conf. Database Syst. Adv. Appl. (DASFAA), 2015, pp. 189–206.

[2] I. Bartolini, P. Ciaccia, and M. Patella, ‘‘SaLSa: Computing the skyline without scanning the whole sky’’, in Proc. Int. Conf. Inf. Knowl. Manage. (CIKM), 2006, pp. 405–414.

[3] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger, ‘‘The R*-tree: An efficient and robust access method for points and rectangles”, in Proc. ACM Special Interest Group Manage. Data (SIGMOD), 1990, pp. 322–331.

[4] S. Borzsony, D. Kossmann, and K. Stocker, ‘‘The skyline operator’’, in Proc. IEEE 17th Int. Conf. Data Eng. (ICDE), Apr. 2001, pp. 421–430.

[5] Z. Chen, H. T. Shen, X. Zhou, and J. X. Yu, ‘‘Monitoring path nearest neighbor in road networks’’, in Proc. ACM Special Interest Group Manage. Data (SIGMOD), 2009, pp. 591–602.

[6] J. Chomicki, P. Godfrey, J. Gryz, and D. Liang, ‘‘Skyline with presorting’’, in Proc. IEEE Int. Conf. Data Eng. (ICDE), Mar. 2003, pp. 717–719.

[7] P. Ciaccia, M. Patella, and P. Zezula, ‘‘M-tree: An efficient access method for similarity search in metric spaces”, in Proc. Int. Conf. Vary Large Data Bases (VLDB), 1997, pp. 426–435.

[8] K. Deng, Y. Zhou, and H. Tao, ‘‘Multi-source skyline query processing in road networks”, in Proc. IEEE Int. Conf. Data Eng. (ICDE), Apr. 2007, pp. 796–805.

[9] A. Guttman, ‘‘R-trees: A dynamic index structure for spatial searching”, SIGMOD Rec., vol. 14, no. 2, pp. 47–57, 1984.

[10] W. T. Hsu, Y. T. Wen, L. Y. Wei, and W. C. Peng, ‘‘Skyline travel routes: Exploring skyline for trip planning”, inProc. IEEE Int. Conf. Mobile Data Manage. (MDM), Jul. 2014, pp. 31–36.

[11] Y.-K. Huang, C.-H. Chang, and C. Lee, ‘‘Continuous distance-based skyline queries in road networks”, Inf. Syst., vol. 37, no. 7, pp. 611–633, 2012.

[12] Y.-K. Huang and Z.-H. He, ‘‘Processing continuous K-nearest skyline query with uncertainty in spatio-temporal databases”, J. Intell. Inf. Syst., vol. 45, no. 2, pp. 165–186, 2015.

[13] X. Huang and C. S. Jensen, ‘‘In-route skyline querying for location-based services”, in Proc. Web Wireless Geograph. Inf. Syst. (W2GIS), 2004, pp. 120–135.

[14] S. Jang and J. Yoo, ‘‘Processing continuous skyline queries in road networks”, in Proc. Int. Symp. Comput. Sci. Appl., Oct. 2008, pp. 353–356.

[15] H.-P. Kriegel, M. Renz, and M. Schubert, ‘‘Route skyline queries: A multipreference path planning approach”, in Proc. IEEE Int. Conf. Data Eng. (ICDE), Mar. 2010, pp. 261–272.

[16] F. Li, D. Cheng, M. Hadjieleftheriou, G. Kollios, and S.-H. Teng, ‘‘On trip planning queries in spatial databases”, in Proc. Int. Symp. Spatial Temporal Databases (SSTD), 2005, pp. 273–290.

[17] S. Pan, Y. Dong, J. Cao, and K. Chen, ‘‘Continuous probabilistic skyline queries for uncertain moving objects in road network”, Int. J. Distrib. Sensor Netw., vol. 2014, Mar. 2014, Art. no. 365064.

[18] Papadias, Y. Tao, G. Fu, and B. Seeger, ‘‘An optimal and progressive algorithm for skyline queries”, in Proc. ACM Special Interest Group Manage. Data (SIGMOD), 2003, pp. 467–478.

[19] C. Sheng and Y. Tao, ‘‘Worst-case I/O-efficient skyline algorithms”, ACM Trans. Database Syst., vol. 37, no. 4, 2012, Art. no. 26.

[20] W. Son, S.-W. Hwang, and H.-K. Ahn, ‘‘MSSQ: Manhattan spatial skyline queries”, Inf. Syst., vol. 40, pp. 67–83, Mar. 2014.

[21] Y. Tian, K. C. K. Lee, and W.-C. Lee, ‘‘Finding skyline paths in road networks”, in Proc. ACM Int. Conf. Adv. Geograph. Inf. Syst. (SIGSPATIAL), 2009, pp. 444–447.

[22] Y.-T. Wen, K.-J. Cho, W.-C. Peng, J. Yeo, and S.-W. Hwang, ‘‘KSTR: Keyword-aware skyline travel route recommendation”, in Proc. IEEE Int. Conf. Data Mining (ICDM), Nov. 2015, pp. 449–458.

[23] B. Yang, C. Guo, C. S. Jensen, M. Kaul, and S. Shang, ‘‘Stochastic skyline route planning under time-varying uncertainty”, in Proc. IEEE Int. Conf. DataEng. (ICDE), Mar./Apr. 2014, pp. 136–147.