ijecs open access www.ijecs.in
*International Journal Of Engineering And Computer Science ISSN:2319-7242*
*Volume 4 Issue 6 June 2015, Page No. 12512-12515*

# An Efficient Trip Planner for Time Dependent Road Networks

***Sithara M P[1], Varsha C[2]***

[1]Department of CS, KMCT College Of Engineering,
Calicut University, Kerala, India
*sithararajan@gmail.com*

[2]Department of CS, KMCT College Of Engineering,
Calicut University, Kerala, India
*Varshachandramohan@gmail.com*

***Abstract: Management of transportation systems has become increasingly important in many real applications such as location-based services, supply chain management, traffic control, and so on. These applications usually involve queries over spatial road networks with dynamically changing and complicated traffic conditions. When we consider road network, route search and optimal path queries are two important types of queries. A path query returns a path that is a set of points that connects the source and destination. The optimal path queries find the optimum path from set information. In the case of road network users give some specification about the travelling with or without constraints. The optimal path queries optimize the possible paths and give the optimal path that satisfies all the constraints. The road network mainly deals with time dependent parameters A spatial road network can be modeled by a large graph in a 2-dimensional geographical space, whose edges correspond to road segments, and are associated with weights related to the traffic information. This paper, mainly focus on finding one of the best path that has minimum travel time. User can select the query points and Candidate plans are generated based on the selected points. To reduce the search space time interval pruning and probabilistic pruning strategies are implemented. Finally the best plan is refined based on a probabilistic threshold.***

**Keywords:** Path queries, Optimal Path, Shortest Distance, and Backward Search.

## 1. Introduction

Standard algorithms for finding shortest path in graph assume that costs are deterministic. The most commonly used algorithm to find the shortest path is Dijkstra's algorithm [1]. The optimal route query processing is mainly used in the road network. A road networks is represented by a large graph in 2-dimentional space. The edge is considered as a road segment. Various queries have been proposed to find the optimal path in road network. The optimal path queries find the optimal path from the given set of information. Various techniques are used for the processing of path queries .Some of the techniques use travelling constraints which are either total order or partial order. Optimal path query processing finds the entire possible path and then refines the best path. Different queries use different algorithms. This paper, mainly focus on finding one of the best path that has minimum travel time. Slightly
modified versions of the standard algorithms will produce optimal results. Two pruning strategies are used to reduce the search space. User can select the query points. Candidate plans are generated for the selected points. To reduce the search space two pruning strategies are implemented. Finally the best plan is refined based on probabilistic threshold.

The paper is organized as follows: section 2 presents various path query algorithms. Section 3 presents the proposed system. Section 4 discuss about the framework of the proposed system .

## 2. Various Path Query Algorithms.

### 2.1 Fast Approximation Algorithm

On Trip Planning Queries are the efficient and exact solutions for the general optimal route queries. A set of query points are given, where each point belongs to a specific category, a starting point S and a destination E. TPQ retrieves best trip that starts at S and passes through at least one point from each category and ends at E[4]. Four algorithms with various approximation ratios are used in terms of $m$ and $\rho$, *where* m is total number of categories and ρ is maximum category cardinality. Two greedy algorithms with tight approximation ratios with respect $m$ are Nearest Neighbor Algorithm and Minimum Distance Algorithm.

Nearest Neighbor Algorithm Iteratively visit the nearest neighbor of last vertex added to the trip from all vertices in categories that have not been visited yet.

Minimum Distance Algorithm is better when compared with Nearest Neighbor Algorithm. Algorithm chooses set of vertices, one vertex per category and sum of cost per vertex is minimum cost among all vertices belong to respective category. Creates trip by traversing these vertices in nearest neighbor order.

### 2.2 New Dijkstra Based Algorithm

A new DIJKSTRA-based algorithm is used to find the optimal LTT with time complexity $O((n \log n + m)\alpha(T))$ and space complexity $O((n + m)\alpha(T))$, where $n$ is the number of nodes, $m$ is the number of edges, and $\alpha(T)$ is the cost required for each function operation. Propose algorithm by decoupling path selection and time refinement. DIJKSTRA-based algorithm is used for time-refinement and a linear-time algorithm for path selection.

Dijkstra Based time refinement: time-refinement means to compute and refine the earliest arrival-time function $gi(t)$ for every node $vi$ in $V$ The earliest arrival time function is calculated for every node. Then refine the arrival time function, incrementally in the given starting time interval. Incrementally

means refine the earliest arrival time function by extending the starting sub-interval to larger starting sub interval.

### 2.3 P*-A Best First Search Algorithm

P* algorithm, a best-first search algorithm for efficient probabilistic path query evaluation. P* carries the similar spirit as the A* algorithm. It iteratively visits the next vertex that is most likely to be an answer path using a heuristic evaluation function, and stops when the rest unexplored paths have no potential to satisfy the query. However, the two algorithms are critically different due to the different types of graphs and queries. A* is used to find the shortest path between two vertices *u* and *v* in a certain graph. Therefore, the heuristic evaluation function for each vertex *vi* is simply the sum of the actual distance between *u* and *vi* and the estimated distance between *vi* and *v*. P* aims to find the paths between two vertices *u* and *v* that satisfy the weight threshold *l* and probability threshold *p* in a probabilistic graph with complex correlations among edge weights. Therefore, the heuristic evaluation functions for each vertex *vi* is the joint distribution on a set of correlated random variables. This posts serious challenges in designing heuristic evaluation functions and calculation. Three heuristic evaluation functions that can be used in the P* algorithm are Constant Estimate, Min-Value Estimate and Stochastic Estimate.

### 2.4 Priority First Search With Dominance Pruning

This is a modified version of shortest path algorithm. Here all the undominated paths are maintained .If one path to a node dominates other then stochastic consistency condition ensures that the second cannot be part of overall shortest path. A method called priority first search with dominance pruning, a variant of priority first search is used. Two data structures are maintained Priority Queue and Closed list. Priority Queue consist of path and path cost. Closed list associates nodes with undominated paths found to that node.

The general procedure is:

Step1: Algorithm first add origin to priority queue with path cost 0.

Step2: Then get highest-priority item from PQ. If this item has lower expected utility than a known path to the destination, then terminate and return the best path to the destination found so far.

Step3: Then add item to closed list. If there is already another path to that node with dominating priority then, go to step 2. Otherwise, add the path and its cost to the closed-list associated with this node.

4. Generate successors to this item. Construct new paths for each possible bus we could take from this node, and put the resulting items on PQ. Go to step 2.

### 2.5 Route Traversal And Link Traversal Search With Transitions

Route Traversal Search traverses nodes similar to DFS. When expanding the current search node, RTS consider all successor nodes for each route that includes this node. It employs a termination check, based on the reachability information within the routes. The principle depends on inverted file R-index on the route collection. Route Traversal Search with Transitions exploits information about the transitions among routes stored in T-index. It employs a stronger termination check based on the transitions between routes. In Link Traversal Search the search stops as soon as LTS visits a node (link) that lies on the same route with the target. Algorithm employs an augmented inverted file on the route collections, termed R-Index$^{+,}$ which associates a node with the routes that contain it and the immediately following link. Link Traversal Search with Transitions enforces a stronger termination check than LTS using the transition graph of the route collection.Finishes when it reaches a node that is closer than two routes away from the target. It uses information from the T –Index.

### 2.6 Backward Search And Forward Search Solution

The backward search methodology computes the optimal routes in reverse order of its points. Two algorithms are developed based on BSS are Simple Backward Search (SBS) and Batch Backward Search [6]

SBS computes an upper bound of the optimal route length, using a greedy algorithm .Then, SBS retrieves the set CS of candidate points that may be part of the optimal route which are those that 1) belong to any category contained in the visit order graph , 2) fall within distance to the query start point q. This can be performed efficiently, e.g., by executing a circular range query on each R-tree that indexes a category of points relevant to the query.

The batch backward search (BBS) method, improves SBS by employing batch processing in the backward join operations. Specifically, both the candidate set CS and the route set is partitioned into clusters before participating in a backward join. The partitioning of CS first groups points by their category, and then for each group, the points are further partitioned into clusters based on their spatial proximity. The partitioning of route set follows a similar strategy, by first grouping routes based on the categories they cover, and then clustering each group according to the locations of their start points. The clustering module in BBS must be highly efficient, since it is called during query time.

The forward search approach traverses the search space in a depth-first manner, and incrementally improves the bound for optimal route length. As an additional benefit, forward search methods report results progressively, i.e., they first quickly produce one solution to the query, and then incrementally update it, until reaching the optimal one or being terminated by the user. Two algorithms developed based on FSS are Simple forward Search (SFS) and Batch Forward Search (BFS)

The simple forward search (SFS) method resembles Greedy in that it also extends the current path by adding the nearest point from an unvisited category. A major difference between the two is that SFS backtracks after it obtains a complete route.

BFS follows the same depth-first search paradigm as SFS. However, instead of enumerating individual routes, BFS searches for sequences of clusters, which we call cluster paths.

Specifically, in a preprocessing step, BFS partitions the candidate set into clusters as in BBS, i.e., the points in each cluster belong to the same category, and are close to each other in space.

## 3. Proposed Method

The proposed method TPQ(Trip Planner Query) retrieves trip plans that traverse a set of query points in PT-Graph having the minimum traveling time with high confidence. TPQ problem considers multiple (≥2) places, uses the PT-Graph with the probabilistic model (rather than a certain graph), and has a different goal of minimizing the traveling time on road networks only (instead of the total time that includes the staying time at vertices).

The spatial road network is modeled by a probabilistic time-dependent graph (PT-Graph). Specifically, a PT-Graph G is a directed graph, in which two connected vertices, $v_i$ and $v_j$, are linked by two bidirectional edges $e_{i,j}$ and $e_{j,i}$. Any edge $e_{i,j}$ in PT-Graph G is associated with an uncertain
edge-delay function (UDF).

A trip planner can specify *n* places of interest to visit (i.e., query points) on the road network, which are denoted as $q_1$, $q_2$, . . . , and $q_n$. For each place qi (1 i n), visitors can stay for some time st(qi) [ST(qi), ST+(qi)], where ST(qi) and ST+ (qi) are the minimum and maximum staying time of travelers at qi. Here, ST (qi) is a constraint, which allows travelers to have enough time to visit qi. Our goal is to find the best plans of the place-visiting order, as well as staying time at each place, such that the total traveling time on road networks (i.e., the total time on the way to targeted places qi) is the smallest with high confidence.

Due to the data uncertainty in the PT-Graph, TPQ problem can be solved by first conducting queries in each possible world of the PT-Graph, and then combining the query results from all the possible worlds, where each possible world is a materialized instance of road network (PT-Graph) with fixed traffic conditions that can appear in reality. However, since the number of possible worlds can be exponential, this method is inefficient, and thus challenging to obtain the best trip plans efficiently from a PT-Graph. Thus, to tackle the efficiency issue of TPQs, effective pruning techniques are used to filter out false alarms of trip plans.

## 4. Framework of Proposed Method

The main framework use in the proposed method is filter and refine framework. The method is divided into three phases: Candidate Generation, Filtering phase and Refinement phase.

### 4.1 Candidate Generation

Here candidate plans are generated for the selected query points. Depth First Search(DFS) approach is used to obtain the candidate plans. A tree is constructed with R as root and query points as tree nodes. From each node different paths are generated in different visiting order. A new tree class is created and traversed in the tree to get all paths that contain the selected query points. For each selected query points the adjacent node and the corresponding edge is generated. The resulting paths are the set of candidate plans. Each candidate plan will contain all the selected query points.

### 4.2 Filtering Phase

In the filtering phase, effective pruning strategies are utilized to filter out false alarms of trip plans. Two pruning strategies are, time interval pruning and probabilistic pruning. The probabilistic pruning considers probabilistic distributions to prune those trip plans with low confidence.

**Time interval pruning** utilizes lower/upper bounds of the traveling times for trip plans to filter out false alarms .

**Probabilistic pruning with β score** considers β score to prune those trip plans with low confidences.

#### 4.2.1 Time interval Pruning

To enable the time interval pruning one critical issue is on how to obtain lower/upper bounds(LBT and UBT)of the traveling time quickly. The basic idea is to infer such time bounds from UDFs of edges, as well as the staying time intervals at vertices, in the PT-Graph.

Each point is associated with their staying time interval and each edge is associated with travel time interval.LBT is calculated based on the minimum stay time and travel time .UBT is calculated based on maximum stay time and travel time. LBT is calculated based on minimum stay time and minimum travel time.

Assume that we have a path $v_1$ -> $v_2$ -> $v_3$ (of length 2), where the visitor departs from v1 at timestamp dep($v_1$) within a
time interval [dep($v_1$), dep+ ($v_1$)]. Note that, if vertex $v_2$ is one of the specified query points, the staying time at $v_2$ is bounded by [st($v_2$), st+($v_2$)] = [ST($v_2$), ST+($v_2$)] otherwise, when vertex $v_2$ is a normal vertex (other than query point) on road networks, the staying time st($v_2$) at $v_2$ is 0, that is, st ($v_2$) = st+ ($v_2$) = 0. Similarly UBT is calculated based on maximum stay time and maximum travel time.

Let threshold τ be the smallest time upper bound, UBT(best plan), among all the candidate plans that have seen so far. Then, the time interval pruning method is to rule out those plans, Plan, whose lower bounds of the traveling time, LBT(Plan), are greater than or equal to threshold (i.e=UBT(best plan)). Then insert the remaining plan to candidate set. In the next step retrieve all
the points that have visited. If their exits other points in the graph, then replace plans with those query points in different orders and generate plan1. Update travelling time interval of finer plans, plan1 and .Then apply filtering to plan1 and insert remaining plans to candidate set. There are still some plans that we have not fully investigated, that is, those plans Plan2 with paths among query points dangling across different index nodes. Therefore, again utilize the time interval to rule out false alarms for these plans, and add those unpruned plans to candidate set.

#### 4.2.2 Probabilistic Pruning With β Score

To enable probabilistic pruning the candidate set after time interval pruning is considered.
Here user can select the day of travel to avoid the traffic jams. β is maximum velocity function. β score is calculated for each candidate plan based on distance, velocity and LBT. To enable probabilistic pruning remove those plans whose value is smaller than the probabilistic threshold. Probabilistic threshold is the average of LBT of all candidate plans.

### 4.3 Refinement

In refinement phase, the remaining candidates after the traversal are refined to return actual TPQ answers. Refining most promising plans from the set of plans in the candidate set. The β value and distance are relatively sorted to get most promising candidate plan.

## 5. Conclusions

Path queries are used to find the optimal path in the road network. Various queries have been proposed. All methods give an optimal path. Some consider partial order constrains while others consider total order constraint. This paper made a detailed survey about various path query methods used in road networks and investigates an important problem, called Trip Planner Query, in the time dependent road network. This query helps travelers to find those trip plans, which visit several places of interest and have minimum traveling time on road network with high confidence. The proposed method takes travel time into consideration and helps the travelers to find trip plans, which visit several places of interest and have minimum travelling time on road network with high confidence. Initially the candidate plans are generated using DFS approach. This will not give an optimal result because there can be exponential number of plans. So to reduce the search space effective pruning strategies are used to prune the plans with less confidence.

## References


[1] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[2] M. Hua and J. Pei, "Probabilistic path queries in road networks:Traffic uncertainty aware path selection," in *Proc. 13th EDBT*,Lausanne, Switzerland, 2010, pp. 347–358.

[3] R. Jin, L. Liu, B. Ding, and H. Wang, "Distance-constraint reachability computation in uncertain graphs," in *Proc. VLDB*, Jun. 2011,pp. 551–562.

[4] F. Li, D. Cheng, M. Hadjieleftheriou, G. Kollios, and S.-H. Teng, "On trip planning queries in spatial databases," in *Proc. 9th Int.Conf. SSTD*, Angra dos Reis, Brazil, 2005.

[5] M. Potamias, F. Bonchi, A. Gionis, and G. Kollios, "*K*-nearest neighbors in uncertain graphs," in *Proc. VLDB*, Singapore, Sep.2010.

[6] Li J, Yang Y.D, Mamoulis N(2013), "Optimal Route Queries with Arbitrary Order Constraints", IEEE Trans. Computers, vol. 25, no. 5, pp. 1097- 1110

[7] Li F, Cheng D, Hadjieleftheriou M, Kollios G, Teng S.H(2005), "On Trip Planning Queries in Spatial Databases", Proc. Ninth Int'l Conf. Advances in Spatial and Temporal Databases (SSTD)

[8]Bin Yang, Chenjuan Guo, Christian S. Jensen, Manohar Kaul, Shuo Shang," Multi-Cost Optimal Route Planning Under Time-Varying Uncertainty"

[9]Xiang Lian and Lei Chen, Member,\Trip Planner Over Probabilistic Time-Dependent Road Networks", IEEE Transactions On Knowledge And Data Engineering, Vol. 26, No. 8, August 2014.

[10] R. Cheng, D. V. Kalashnikov, and S. Prabhakar, Evaluating probabilistic queries over imprecise data", in Proc. SIGMOD, SanDiego, CA, USA, 2003, pp. 551562.

[11] D. Papadias, J. Zhang, N. Mamoulis, and Y. Tao, \Query processing in spatial network databases",in Proc. 29th VLDB, Berlin, Germany, 2003.

[12] J. Pei, B. Jiang, X. Lin, and Y. Yuan, "Probabilistic skylines on uncertain data", in Proc. 33rd VLDB, 2007.

[13] M. L. Yiu, D. Papadias, N. Mamoulis, and Y. Tao, "Reverse nearest neighbors in large graphs", in Proc. ICDE, 2005.