ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 7, Issue 4, April 2019

Route Search on Road Networks Using User Clues

A M Jagat[1], Anurag Ramesh[2], Rajat Jangra[3], Ziaur Rahman[4] , Shankar Rana[5]

Student, Department of Computer Science and Engineering, Sambhram Institute of Technology, Bengaluru, India[1]

Student, Department of Computer Science and Engineering, Sambhram Institute of Technology, Bengaluru, India[2]

Student, Department of Computer Science and Engineering, Sambhram Institute of Technology, Bengaluru, India[3]

Student, Department of Computer Science and Engineering, Sambhram Institute of Technology, Bengaluru, India[4]

Assistant Professor, Department of Computer Science and Engineering, Sambhram Institute of Technology, Bengaluru, India[5]

ABSTRACT: With the advancement in geo-positioning technologies and location-based services, it is nowadays common for road networks to have textual contents on the vertices. Previous work on finding an optimal route that covers a sequence of query keywords has been studied in recent years. However, in many practical scenarios, an optimal route might not always be desirable. For example, a personalized route query is issued by providing some clues that describe the spatial context between PoI's along the route, where the output can be far from the optimal one. Therefore, in this paper, we investigate the problem of clue-based route search (CRS), which allows a user to provide clues on keywords and spatial relationships. First, we propose a greedy algorithm and a dynamic programming algorithm as baselines. To improve efficiency, we implement a branch-and-bound algorithm that prunes unnecessary vertices in query processing. In order to quickly locate candidate, we propose an AB-tree that stores both the distance and keyword information in tree structure. For reducing the index size, we construct a PB-tree by utilizing the virtue of 2-hop label index to pinpoint the candidate. Extensive experiments are performed and verify the superiority of our algorithms and index structures.

KEYWORDS – Spatial keyword queries, clue, Point-of-Interest, travel route search, query processing.

I. INTRODUCTION

With the further development of location-based services and geo positioning technologies, there is a clear trend that an increasing amount of geo-textual objects are available in many applications. For example, the location information as well as concise textual descriptions of some businesses (e.g., restaurants, hotels) can be easily found in online local search services (e.g., yellow pages). To provide great user experience, various keywords related to the spatial query models and techniques have emerged such that the geo-textual objects can be efficiently retrieved. It is common to search a Point-of-Interest (PoI) by providing exact address or distinguishable keyword in a region which can uniquely pinpoint the location. For example, we type the address "73 Mary St, Brisbane" or the name "Kadoya" on Google Maps to find a Japanese restaurant in the CBD area. Some existing work [8], [15], [26], [31], [33], [35] extends such query to more sophisticated settings, such as retrieving a group of Geo textual objects (usually more than 2) or a trajectory covering multiple keywords. However, it is not uncommon that a user aims to find a PoI with less distinguishable keyword such as "restaurant", but she can only provide more or less spatial textual context information around the PoI. Liu et al. [25] formalize such context information as clues and use them to identify the most promising PoI's. Different with their work, we target to find a feasible route on road networks by using clues. Particularly, in this paper, we investigate a novel query type, namely clue-based route search (CRS), which allows a user to provide clues on textual and spatial context along the route such that a best matching route w.r.t. the clues is returned. More specifically, a CRS query is defined over a road network G, and the input of the query consists of a source vertex vq and a sequence of clues, where each clue contains a query keyword and a user expected network distance. A vertex contains a clue keyword is considered as a match vertex. The query returns a path P in G starting at vq, such that (i.) P passes through a sequence of match vertices (PoI's) w.r.t. the clues and (ii.) the network distances between two contagious matched vertices are close to the corresponding user specified distance such that the user's search intention is satisfied**.**

II. LITERATURE SURVEY

1. Hierarchical hub labelling's for shortest paths

As Abraham, D. Delling stated, we study hierarchical hub labelling's for computing shortest paths. Our new theoretical insights into the structure of hierarchical labels lead to faster pre-processing algorithms, making the labelling approach practical for a wider class of graphs. We also find smaller labels for road networks, improving the query speed.

2. Fast shortest path distance queries on road networks by pruned highway labelling

T. Akiba, Y. Iwata stated, we propose a new labeling method for shortest-path and distance queries on road networks. We present a new framework (i.e. data structure and query algorithm) referred to as highway-based labeling and a preprocessing algorithm for it named pruned highway labeling. Our proposed method has several appealing features from different aspects in the literature. Indeed, we take advantages of theoretical analysis of the seminal result by Throop for distance oracles, more detailed structures of real road networks, and the pruned labeling algorithm that conducts *pruned*Dijkstra's algorithm. The experimental results show that the proposed method is comparable to the previous state-of-the-art labeling method in both query time and in data size, while our main improvement is that the preprocessing time is much faster.

3. Dynamic and historical shortest path distance queries on large evolving networks by pruned landmark labelling

As T. Akiba, Y. Iwata stated, we propose two dynamic indexing schemes for shortest-path and distance queries on large time-evolving graphs, which are useful in a wide range of important applications such as real-time network-aware search and network evolution analysis. To the best of our knowledge, these methods are the first practical exact indexing methods to efficiently process distance queries and dynamic graph updates. We first propose a dynamic indexing scheme for queries on the last snapshot. The scalability and efficiency of its offline indexing algorithm and query algorithm are competitive even with previous static methods. Meanwhile, the method is dynamic, that is, it can incrementally update indices as the graph changes over time. Then, we further design another dynamic indexing scheme that can also answer two kinds of historical queries with regard to not only the latest snapshot but also previous snapshots. Through extensive experiments on real and synthetic evolving networks, we show the scalability and efficiency of our methods. Specifically, they can construct indices from large graphs with millions of vertices, answer queries in microseconds, and update indices in milliseconds.

4. Collective spatial keyword querying

X. Cao, G. Cong, with the proliferation of geo-positioning and geo-tagging, spatial web objects that possess both a geographical location and a textual description are gaining in prevalence, and spatial keyword queries that exploit both location and textual description are gaining in prominence. However, the queries studied so far generally focus on finding individual objects that each satisfies a query rather than finding groups of objects where the objects in a group collectively satisfy a query. We define the problem of retrieving a group of spatial web objects such that the group's keywords cover the query's keywords and such that objects are nearest to the query location and have the lowest inter-object distances. Specifically, we study two variants of this problem, both of which are NP-complete. We devise exact solutions as well as approximate solutions with provable approximation bounds to the problems. We present empirical studies that offer insight into the efficiency and accuracy of the solutions.

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 7, Issue 4, April 2019

5. The multi-rule partial sequenced route query

H. Chen, W.-S. Ku, about trip planning search (TPS) represents an important class of queries in Geographic Information Systems (GIS). In many real-world applications, TPS requests are issued with a number of constraints. Unfortunately, most of these constrained TPS cannot be directly answered by any of the existing algorithms. By formulating each restriction into rules, we propose a novel form of route query, namely the multi-rule partial sequenced route (MRPSR) query. Our work provides a unified framework that also subsumes the well-known trip planning query (TPQ) and the optimal sequenced route (OSR) query. In this paper, we first prove that MRPSR is *NP*-hard and then present three heuristic algorithms to search for near-optimal solutions for the MRPSR query. Our extensive simulations show that all of the proposed algorithms can answer the MRPSR query effectively and efficiently. Using both real and synthetic datasets, we investigate the performance of our algorithms with the metrics of the route distance and the response time in terms of the percentage of the constrained points of interest (POI) categories. Compared to the LORD-based brute-force solution, the response times of our algorithms are remarkably reduced while the resulting route length is only slightly longer than the shortest route.

III. RELATED WORK

A. Top-k Spatial Keyword Search

Searching geo-textual objects with query location and keywords has gained increasing attention recently due to the popularity of location-based services. In Euclidean space, IR2-tree [13] integrates signature files and R-tree to answer Boolean keyword queries. IR-tree [12] is an R-tree augmented with inverted files that supports the ranking of objects based on a score function of spatial distance and text relevancy. Cao et al. [7] proposes a location-aware top-k prestige-based text retrieval (LkPT) query, to retrieve the top-k spatial web objects ranked according to both prestige-based text relevance (PR) and location proximity. [10] Provides an all-round survey of 12 state-of-art geo-textual indices and proposes a benchmark that enables the comparison of the spatial keyword query performance. Zhang et al. [31], [32] proposes the m closet keyword query (mCK query) which aims to find the closest objects that match the query keywords and their distance diameter is minimized. Studies the problem of direction aware spatial keyword search, which aims at finding the k nearest neighbours to the query that contain all input keywords and satisfy the direction constraint. Rocha et al. [27] address the problem of processing top-k spatial keyword queries on road networks where the distance between the query location and the spatial object is the length of shortest path. ROAD [21] organizes the road network as a hierarchy of sub-graphs, and connects them by adding shortcuts. For each sub graph, an object abstract is generated for keyword checking. By using network expansion, the sub graphs without intended object are pruned out. G-tree [36] adopts a graph partitioning approach to form a hierarchy.

B. Travel Route Search

The travel route search problem has been substantially studied for decades. Traveling Salesman Problem (TSP) [11] is the most classic problem in route planning. TSP aims to find the round trip that has the minimum cost from a source point to a set of targets. Li et al. [22] study the problem of Trip Planning Query (TPQ) in spatial databases, where each object is associated with a location and a category. With a starting point S, a destination E and a set of categories C, TPQ retrieves the best trip that starts at S passes through at least one point from each category, and ends at E. TPQ can be considered as a generalization of Travelling Salesman Problem (TSP), thus two approximation algorithms are proposed. [28] Studies the problem of optimal sequenced route (OSR), which aims to find a route of minimum length starting from a source point and passing through a number of typed locations in a specific sequence imposed on the types of the locations. They propose a LORD and R-LORD algorithms to filter out the locations that cannot be in the optimal route, thus improves the sequence route (MRPSR), which aims to find an optimal route with minimum distance under some partial category order rules defined in the query. They propose three heuristic algorithms to search for near-optimal solutions for the MRPSR query. [20] proposes a greedy algorithm to find a route whose length is smaller than a specified threshold while the total text relevance of this route is maximized.

IV. PROBLEM DEFINITION

Definition 1 (Clue). A clue is defined as μ(w, d, s), where w is a query keyword, d is a user defined distance, and s [0, 1] is a confidence factor.
Definition 2 (Match). Given a source vertex u and a clue μ(w, d, s), we say that the vertex pair σ(u v) is a match w.r.t. clue μ, if the vertex v contains clue keyword w and the network distance between u and v is in [d(1 − s), d(1 + s)], i.e., $w \in \Phi(v)$ and $dG(u, v) \in [d(1 - s), d(1 + s)]$..

We adopt the idea of distance oracle to calculate the network distance between two input vertices. Given a source-target pairof vertices, returns the shortest network distance between them. As we know, the algorithms and data structures on have been extensively studied by existing works, which can be roughly summarized into two categories, expansion-based methods and lookup-based methods. The most famous expansion-based method for is Dijkstra's algorithm [14], which, given a s-t pair in road network tt, traverses the vertices in tt from s to t. However, the problem of using Dijkstra's algorithm is that it must visit every vertex that is closer to s, and the number of such unneeded.

1 GREEDY CLUE SEARCHALGORITHM

We develop a greedy algorithm as a baseline for answering the CRS query, which is called Greedy Clue Search (GCS) algorithm. Given a query Q = (vq; C), we first add vq into a candidate path. Then we use the Procedure findNextMin() to determine the next match vertex v1 that the matching distance between _1 and _1(vq ! v1), i.e., dm(_1; _1), is reduced. Afterwards, we insert v1 into the candidate path, and continue to find its contagious candidate by findNextMin(). This process is repeated until all the match vertices are firm, thus the candidate path forms a feasible path, denoted as FPvq If we assume Procedure findNextMin() costs time f, then the time complexity of GCS is O(k _ f).

2 CLUE-BASED DYNAMIC PROGRAMMING ALGORITHM

As we know, even though GCS has a short response time, the accuracy of the answer cannot be guaranteed. To achieve better accuracy, we propose an exact algorithm, called Clue-based Dynamic Programming (CDP), to answer the CRS query. Generally, it is challenging to develop an efficient exact algorithm for CRS queries, since we cannot avoid thorough search for PoIs in road networks. For instance, the number of vertices that contain keyword wi 2 C is denoted as jVwi j, thus the time complexity of the brute-force approach, which attempts all possible combinations, is O(Qwi2CjVwi j).

Algorithm 1: Procedure findNextMin()

Input: Source vertex u and clue μ (w; d; ϵ)
Output: min { dm(μ;σ) } and match vertex v

```
1   From u, do network traversal;
2   if a match vertex v is found then
3       dG ← the network distance between u and v;
4       while true do
5           Find next v'  contains w, thus obtain d'G;
6       ifdG< d and d'G> d then
7               break;
8       else
9                   v ←v'  anddG←d'G;
10   return min { dm (μ; σ ) } and v;
```

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 7, Issue 4, April 2019

In CDP, we construct a keyword posting list for each keyword w, which is a list of vertices that contain w. When a CRS query is issued, we sort the posting lists according to the keyword order of wi 2 C. Note that the order of the vertices within each posting list does not matter and can be random, hence are sorted by vertex id for simplicity. It is easy to see that these posting lists actually construct a k-bipartite graph G0, which in fact shows all feasible paths for a given C. The weight of each edge in G0 is computed as the matching distance. Specifically, for each u 2 Vwi ,we define D(wi; u) to denote the minimum identical distance one can achieve with a walk that passes the keywords from w1 to wi consistent with the order in C and stops at u. In other words, the weight of vertex u 2 G0 is computed by D(wi; u), which is the minimum matching distance of all partial feasible paths end at u.
Then we compute D(wi; u) by the following recursive formula:
(i) i = 1: for match vertices u 2 Vw1 , we have

$D(wi, u) = dm\ (\ \mu i(wi, di), \sigma\ (vq \dashrightarrow u))$

(ii) i> 1: for match vertices v ϵ Vwi-1 and u ϵ Vwi , we have
$D(wi; u) = \min_{v\ \epsilon\ Vwi-1}\{\ \max\ \{D(wi-1; v), dm(\mu i, \sigma\ (v \rightarrow u))\}\}$

Algorithm 2: Clue-based Dynamic Programming CDP
Input: Q = (vq, C = { (w1; d1); ……...,(wk, dk) })
Output:FPcdp with dm(C,FPcdp)

```
1      for each u ϵ Vw1 do
2        Initial D(w1, u);
3      for 1 < i <= k do
4        foreach u ϵVwi,do
5        Initial intermediate vector iv(u);
6        for each v ϵVwi-1 do
7            if dm(μ I,σ (v → u)) < D(wi-1, v) then
8                iv(u) insert D(wi-1, v);
9        else
10               iv(u) insert dm(μ i,σ (v→u));
11       D(wi, u)← min { iv(u) }
12 Find min { D(wk, u) };
13 returnFPcdp and dm(C,FPcdp) ←min { D(wk, u) }
```

3 BRANCH AND BOUND ALGORITHM

Although CDP provides an exact solution, the search efficiency cannot be maintained. For instance, consider the worst case, we assume that all vertices contain query keywords, then the time is O(k _ jV j2). To propose a more efficient algorithm, we assume there is an artificial directed graph G0, which is similar to the k-bipartite graph in CDP that formed by all candidate vertices containing keywords in C, where the edge of G0 is a match of one clue and in the meantime its direction complies the keyword order of the clue. Note that, G0 is organized into k levels, and each level i corresponds to each keyword wi. Based on G0, we develop a Branch-and-Bound (BAB) algorithm to search G0 in a depth-first manner by applying the filter-and-refine paradigm, which only visits a small portion of vertices in G0. Fortunately, we can use the result of GCS to speed up the search process since it can serve as an initial upper bound.

3.1 Algorithm Outline

We start the searching from level 1 to k to obtain a feasible path FP, if the matching distance dm(C;FP) is greater than the current upper bound, we continue to search for the next candidate feasible path, otherwise we update the upper bound. It is worth noting that it is not necessary to go through every candidate feasible path. If the matching distance at intermediate level already exceeds the upper bound, it can be removed. This process terminates when the

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 7, Issue 4, April 2019

matching distance next to be processed at level 1 can be filtered, since it is impossible to find a feasible path with smaller match distance.

Candidate feasible path updating.

Initially, we keep a stack to store the partial candidate path, which contains a sequence of vertices and corresponding matching distances. First, we fetch vq into the stack, then we continue to find next candidate at level 1. Basically, the key component of this algorithm is to quickly locate the next best match vertex, and the details of Procedure findNext() will be introduced later. Given a partial candidate path FP(vq; v1; : : : ; vi) obtained at level i, we apply findNext() to find the next candidate vi+1 at level i+1. Once vi+1 is found, we compute di+1m (vi+1) which denotes the matching distance at level i + 1 resulted by vi+1, and compare it with current UB. Note that, vi+1 is accepted as a candidate and inserted into the stack if and only if its matching distance di+1 m (vi+1) is smaller than UB. Otherwise, vi is removed from the stack as well as di m(vi). In other words, vi is not valid that the path FP(vq; v1; : : : ; vi□1) cannot survive by passing vi, then we have to find an alternative v0 . As we know vi is the current best candidate at level i, therefore we have to relax the matching distance by finding v0i where dim(vi) _ di m(v0i) and dim(v0i) is less among all the rest vertices untouched at level i. Afterwards, if v0 I is valid, we continue to apply findNext() on it.Upper bound updating. Specifically, after we obtain a feasible path FP(vq; v1; : : : ; vk□1) at level k □ 1, if vk is returnedby findNext(), then we check if dk m(vk) exceeds UB. If vk is not valid, we prune vk and simply repeat the above process.

Algorithm 3: Branch and Bound BAB

```
Input: Q = (vq; C)
Output:FPbab with dm(C;FPbab)
1 Initialize stackV, stackD, and search threshold _;
2 Push vq into stackV;
3        while stackV is not empty do
4         i   stackV.size();
5        iffindNext(vi□1; di;wi; _) = true then
6                Obtain vi and dim(vi);
7                Ø←0:0;
8                Push vi into stackV and dim(vi) into stackD;
9                if i equals to k then
10               max{stackDg}<= UB then
11                       Update UB by maxfstackDg;
12                        Update FPbab by stackV;
13               Update _ by top of stackD;
14               Update stackV and StackD;
15       else
16       Update _ by top of stackD;
17       Update stackV and StackD;
18 returnFPbab and dm(C;FPbab)   UB;
```

V. CONCLUSION AND FUTURE DIRECTIONS

In this paper, we study the problem of CRS on road networks, which aims to find an optimal route such that it covers a set of query keywords in a given specific order, and the matching distance is minimized. To answer the CRS query, we first propose a greedy clue-based algorithm GCS with no index where the network expansion approach is adapted to greedily select the current best candidates to build feasible paths. Then, we devise an exact algorithm, namely clue-based dynamic programming CDP, to answer the query that enumerates all feasible paths and finally returns the optimal result. To further reduce the computational overhead, we propose a branch-and-bound algorithm

BAB by applying filter-and-refine paradigm such that only a small portion of vertices are visited, thus improves the search efficiency. In order to quickly locate the candidate vertices, we develop AB-tree and PB-tree structures to speed up the tree traversal, as well as a semi active index updating mechanism. Results of empirical studies show that all the proposed algorithms are capable of answering CRS query efficiently, while the BAB algorithm runs much faster, and the index size of PB-tree is much smaller than AB-tree. Several directions for future research are promising. First, users may prefer a more generic preference model, which combines PoI rating, PoI average menu price, etc, in the query clue. Second, it is of interest to take temporal information into account and further extend the CRS query. Each PoI is assigned with an opening hours' time interval [To; Tc], and each clue contains a visiting time t, where the resulting query aims to find a path such that the time interval of each matched PoI covers the visiting time. Third, requiring users to provide exact keyword match is difficult sometimes as they are just providing "clue", which may be imprecise in nature. Thus, it is of interest to extend our model distance can be modified by incorporating both spatial distance and textual distance together through a linear combination.

REFERENCES

[1] I. Abraham, D. Delling, A. V. Goldberg, and R. F. Werneck. Hierarchical hub labelings for shortest paths. In ESA, pages 24–35. Springer, 2012.
[2] T.Akiba, Y. Iwata, K.-i. Kawarabayashi, and Y. Kawata. Fast shortestpath distance queries on road networks by pruned highway labeling. In ALENEX, pages 147–154. SIAM, 2014.
[3] T. Akiba, Y. Iwata, and Y. Yoshida. Fast exact shortest-path distance queries on large networks by pruned landmark labeling. In SIGMOD, pages 349–360. ACM, 2013.
[4] X. Cao, L. Chen, G. Cong, and X. Xiao.Keyword-aware optimal route search. PVLDB, 5(11):1136–1147, 2012.
labeling. In WWW, pages 237–248. ACM, 2014.
[5] H. Chen, W.-S.Ku, M.-T.Sun, and R. Zimmermann.The multi-rule partial sequenced route query. In SIGSPATIAL, page 10. ACM, 2008.
[6] X. Cao, L. Chen, G. Cong, and X. Xiao.Keyword-aware optimal route search. PVLDB, 5(11):1136–1147, 2012.
[7] X. Cao, G. Cong, and C. S. Jensen.Retrieving top-k prestige-based relevant spatial web objects.PVLDB, 2010.
[8] X. Cao, G. Cong, C. S. Jensen, and B. C. Ooi.Collective spatial keyword querying. In SIGMOD, pages 373–384. ACM, 2011.
[9] H. Chen, W.-S.Ku, M.-T.Sun, and R. Zimmermann.The multi-rule partial sequenced route query. In SIGSPATIAL, page 10. ACM, 2008.
[10] L. Chen, G. Cong, C. S. Jensen, and D. Wu. Spatial keyword query processing: an experimental evaluation. PVLDB, 2013.
[11] N. Christofides. Worst-case analysis of a new heuristic for the travelling salesman problem.Technical report, DTIC Document, 1976.
[12] G. Cong, C. S. Jensen, and D. Wu. Efficient retrieval of the top-k most relevant spatial web objects. PVLDB, 2009.
[13] I. De Felipe, V. Hristidis, and N. Rishe. Keyword search on spatial databases.In ICDE, 2008.
[14] E. W. Dijkstra. A note on two problems in connexion with graphs.Numerischemathematik, 1(1):269–271, 1959.