2016 IEEE 18th International Conference on High Performance Computing and Communications; IEEE 14th International Conference on Smart City; IEEE 2nd International Conference on Data Science and Systems

Searching Time Period-based Longest Frequent Path in Big Trajectory Data

Xu Gao *†, Limin Guo ‡
Guangyan Huang§, Zhiming Ding ‡✉
*University of Chinese Academy of Sciences, Beijing, China
†Institute of Software, Chinese Academy of Sciences, Beijing, China
gaoxu,yanjun@nfs.iscas.ac.cn
‡Beijing University of Technology, Beijing, China
{guolimin, zhiming, juncheng}@bjut.edu.cn

Yanjun Wu †, Borui Cai ‖
Xiangxi Meng ¶, Juncheng Chen ‡
§School of Information Technology, Deakin University, Melbourne, Australia
guangyan.huang@gmail.com
¶Beihang University, Beijing, China
eeydxm@sina.com
‖Xilinx, inc., Beijing, China
Borui.cai@hotmail.com

***Abstract*—Trajectories contain considerable routing information, and trajectory-based routing gains the attention of researchers recently. This paper argues the problem of searching time period Longest Frequent Path(TPLFP). The TPLFP, as one of the reasonable alternative Most Frequent Path (MFP), is close to MFP enough and maximizes the number of frequent route segments. First, we define TPLFP formally. To acquire accurate path frequencies, we describe the footmark, the footmark graph notations and design the Linear Sketch Footmark Index (LSFI) to speed up extracting proper footmark and constructing related footmark graph. Next, we develop a Best-First search algorithm with four pruning strategies. Next, we give an advanced footmark graph and its building algorithm. The extensive experiments demonstrate the effectiveness and efficiency of our index schemes and algorithms, which can find TPLFP results in expected response time.**

1. Introduction

The continued proliferation of mobile devices equipped with a positioning module (e.g., GPS, GSM, WIFI) produces increasing volumes of trajectoris, which contain routing details. Path planning, also known as finding the most desirable path, plays a crucial role in location based services. Through history trajectories, however, we can obtain new paths otherwise shortest, fastest ones. Also, new paths are diversified w.r.t different criteria including popularity, frequency and personal preferences, etc.

Most frequent or popular path is the path whose segments are travelled most, while time period longest frequent path is the path whose segments are travelled frequently and the number of segments which is frequent enough is largest in a tiem period. As a alternative most frequent path, TPLFP, which provides users with extra routing options other than the shortest/fastest path, is very useful in many real world applications. In vehicle navigation systems, .

Figure 1 shows a road network whose line width indicates the edge frequencies(the broader the more frequent).

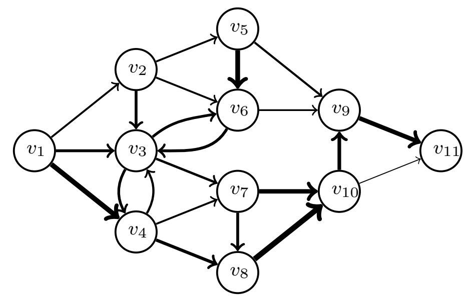


Figure 1: Road network G

Considering routes between v_1 and v_{11}, the number of Least Route Segment paths(with 4 segments) is 6. It is clear that $P_1 = v_1 \to v_2 \to v_6 \to v_9 \to v_{11}$ has only one frequent road edge, however $P_2 = v_1 \to v_4 \to v_8 \to v_{10} \to v_{11}$ has two more frequent road segments. In path P_2 there is a edge from v_{10} to v_{11} with lowest frequency in the road network, and this make people seldom select it. As an alternative to P_2, $P_3 = v_1 \to v_4 \to v_8 \to v_{10} \to v_9 \to v_{11}$ has one more edges than P_2, however, all the passed edges except for $v_4 \to v_8$ are very frequent. Therefore, if the distance, the volumes of consumed fuel or time constraint are acceptable, P_3 is a reasonable path which passing one more edge but with more frequent edge. Then, P_3 is called the Longest Frequent Path, which maximizes the number of very frequent edges while meets constraint of frequent cost.

The objective of TPLFP is to find the time period longest frequent path, whose frequency is near the most frequent path's while maximizing the number of frequent enough route segments. As one kind of alternative most frequent path, TPLFP has a huge searching space as the alternative shortest path does.

This paper study the problem of longest frequent paths, and there are some contributions as follows:

- provides and formalizes the new problem of TPLFP;
- gives a new trajectory index, footmark graph, advanced

978-1-5090-4297-5/16 $31.00 © 2016 IEEE
DOI 10.1109/HPCC-SmartCity-DSS.2016.176

footmark graph and their construction algorithms;

- designs a Best-First search algorithm and pruning strategies;
- performs experiments on a real dataset, evaluates the effectiveness and efficiency of index and algorithms.

The remaining parts of this paper are structured as: Section 2 reviews related works. Section 3 defines TPLFP formally. Section 4 presents footmark graph and its generation algorithm. Section 5 gives the Best-First search algorithm for TPLFP. Additionally, section 6 provides advanced footmark graph. Section 7 discusses the comprehensive experiments. Finally, Section 8 concludes the paper.

2. Related Work

There are studies about path planning, such as shortest, fastest, latest departure or arriving paths [2], [7], [9], [10], [11], [18]. The real driving path, however, is not the shortest or fastest, but rather alternative path constraints by some criteria mostly (e.g., distance,time, keywords, frequency, fuel consumption).

The distance constraint alternative path called alternative shortest path, whose distance is no more than a threshold of the distance of shortest path(e.g., 1.5). Additionally, each sub-path, whose distance is $1/3$ of the total path, is the shortest path between the sub-path's start and destination vertices. This kind of path planning has been proved to be NP-hard [1], [14].

Towards keywords constraint alternative path, MAKR [20] study the problem of multi-approximate keyword routing in the road network. Give multiple query words, MAKR finds candidate paths with points, whose keywords approximate and cover all of the query words, then determines the final path, whose distance is minimized among the candidates. MARK is proved to be **NP** hard. KORS [3] efficiently answers the KOR queries, which is to find a route such that it covers a set of user-specified keywords, a specified budget constraint is satisfied, and an objective score of the route is optimized. Zhang et al. [21] studies the problem of diversified spatial keyword search on road networks which considers both the relevance and the spatial diversity of the results.

Another kind of alternative path with multiple constraints has multiple properties, in which linear weight aggregation is used to convert the multi-property cost into single cost function. Different properties represent different meanings. However, the cost of linear weight aggregation cost is too complicated to understand.

In order to resolve the linear weight aggregation problem, Kriegel et al. [12] provides skyline route queries considering multiple preferences like distance, driving time, the number of traffic lights, gas consumption, etc. Shekelyan et al. [16] introduces multi-criterion linear path skyline queries. A linear skyline path is the subset of the conventional path skyline where the paths are optimal under a linear combination of their cost values.

Popular route searching and frequent path finding are closer related work to the TPLFP problem. MPR [5] is the first to study the common routing preferences of the past travelers and to discover the most popular route between two locations. However, this method tends to favor the paths with fewer vertices. RICK [19] aims at finding top-k popular routes from uncertain trajectories, by extracting route network from these trajectories. TPMFP [13] studies the most frequent path during a given time period between two locations by observing the traveling behaviors of many previous users. However, when the distance between the two places is too further and the interval time span is too smaller, few history trajectories are passing the locations at the same time, thus, provided algorithm will fail.

Another related problem to our work is the management of trajectory data. The famous works include MV3R-tree [17], TB-tree [15], SETI [4], FNR-tree [8] and MON-tree [6]. FNR-tree and MON-tree are for trajectories on the road network, while the inputs of TPLFP are sequences of vertex-timestamp tuples; thus, current trajectory access methods do no help for TPLFP.

3. Problem Statement

3.1. Basic Definition

In this section, we introduce notations used in this paper, then define the problem of TPLFP.

Definition 3.1 (**Road Network**). *Road network is a directed graph $G = (V, E)$, where V is a set of vertices and E is a set of edges. In addition, given $\forall v_i \in V$, $v_i.loc$, represented by longitude and latitude, is the location of vertex v_i, and $v_i.id$ is the identifier for v_i.*

Definition 3.2 (**Path and Trajectory**). *Path $P = x_1 - x_k$ is a non-empty graph $P = (V_p, E_p) \subset G$ therein, where $V_p = \{x_1, x_2, \ldots, x_k\}$, $E_p = \{(x_1, x_2), \ldots, (x_{k-1}, x_k)\}$ and $v_i \neq n_j$ for all $i \neq j$. Trajectory Y is a sequence: $Y = ((x_1, t_1), \cdots, (x_k, t_k))$, which forms a path $x_1 \rightarrow, \cdots, \rightarrow x_k$ on G, where t_i indicates the timestamp when Y passes vertex x_i.*

We use $P.s, P.d, P[i]$ to denote the starting vertex x_1, destination vertex x_k and the ith vertex of P respectively. Also, P can be described as the form of $x_1 \rightarrow x_2 \rightarrow, \ldots, \rightarrow x_k$. Similar to path, $Y.s, Y.d, Y[i], Y[i].t$ describe the starting vertex, destination vertex, the ith vertex and the timestamp when Y passes vertex x_i respectively.

Definition 3.3 (**Footmark**). *Given input Ω, $\forall Y \in \Upsilon$, Y' is a segment of Y, then Y' is a footmark in G, noted as $Y'_{(v_s, v_d, T, \delta)}$, iff 1) $\forall v_i \in Y'$, v_i falls in $\delta MBR(v_s, v_d, \delta)$ and 2) $[Y'.t_s, Y'.t_d] \subseteq T$ hold together, where Υ is the set of all history trajectories, v_s, v_d are the starting vertex and destination vertex respectively, T is the querying time interval, and $\delta \geq 0 \wedge 0 \leq \varepsilon \wedge \theta > 0$ holds.*

In the footmark definition, δMBR(v_s, v_d, δ) is the minimum bounding box w.r.t v_s, v_d, δ, and is called vertices v_s and v_d's δ-expanding minimum bounding box. In detail, δMBR(v_s, v_d, δ) is the minimum bounding box of the circle,

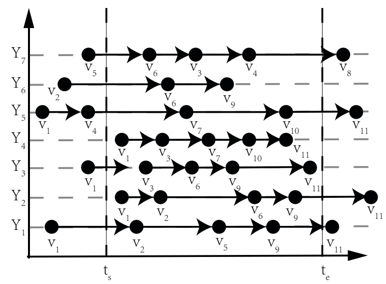


Figure 2: Footmark w.r.t $\Omega = (G, \Upsilon, [t_s, t_e], v_1, v_{11}, 0.5)$

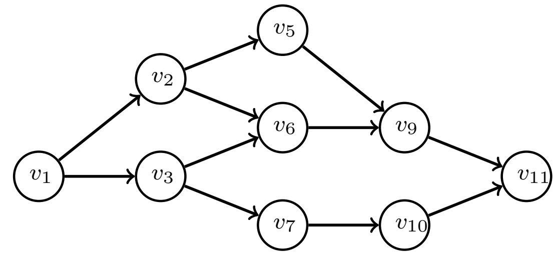


Figure 3: Footmark w.r.t $\Omega = (G, \Upsilon, [t_s, t_e], v_1, v_{11}, 1.5)$

which takes the middle position of line $[v_s, v_d]$ as center, and takes the length of $(1+\delta)|v_s, v_d|$ as the diameter, where $|,|$ computes the Euclidean distance between two vertices.

For instance, Figure 2 shows the footmark set $Y'_{(v_s, v_d, [t_s, t_e], \delta)}$, which contains seven trajectories which taking $v_s = v_1$ and $v_d = v_{11}$ as starting and destination respectively. The footmarks include:

- $Y'_1 = v_2 \rightarrow v_5 \rightarrow v_9$
- $Y'_2 = v_1 \rightarrow v_2 \rightarrow v_6 \rightarrow v_9$
- $Y'_3 = v_3 \rightarrow v_6 \rightarrow v_9 \rightarrow v_{11}$
- $Y'_4 = v_1 \rightarrow v_3 \rightarrow v_7 \rightarrow v_{10} \rightarrow v_{11}$
- $Y'_5 = v_7 \rightarrow v_{10}$
- $Y'_6 = v_6 \rightarrow v_9$
- $Y'_7 = v_6 \rightarrow v_3 \rightarrow v_4$

We should notice that every vertex v_i is not an element of footmark when $Y[i].t \notin T = [t_s, t_e]$. And in the sequel, we simply use Y' to denote $Y'_{(v_s, v_d, T, \delta)}$ when the context is clear. Additionally, $Y'(v_s, v_d, T, \delta)$ contains following three kinds of trajectories: 1) trajectories $Y'_{v_s *}$ pass v_s in T; 2) trajectories $Y'_{* v_d}$ pass v_d in T; 3) trajectories $Y'_{* \backslash \{v_s *, * v_d\}}$ do not pass v_s or v_d.

Thus, $Y' = Y'_{v_s *} \cup Y'_{* v_d} \cup Y'_{* \backslash \{v_s *, * v_d\}}$. Figure 2 shows that $Y'_{(v_1, v_{11}, [t_s, t_e], 0.5)}$, including footmarks from Y'_1 to Y'_7, can be expressed by the union of sets $Y'_{v_1 *} = \{Y'_2, Y'_4\}$, $Y'_{* v_{11}} = \{Y'_3, Y'_4\}$ and $Y'_{* \backslash \{v_1 *, * v_{11}\}} = \{Y'_1, Y'_5, Y'_6, Y'_7\}$.

Definition 3.4 (**Normalized Frequency**). *Given Y'_{ef} and $\forall (u,v) \in E$, $F(u,v)$, as the frequency of edge (u,v), is the number of footmark passing (u,v). As defined in Equations 1, 2 and 3. $F'(u,v)$ is the normalized frequency of (u,v), and f_{max}, f_{min}, is the maximum and minimum frequency in the road network respectively. In the following, We shorted normalized frequency as frequency or edge weight if there is no ambiguity.*

$$F'(u,v) = \frac{f_{max} - F(u,v) + 1}{f_{max} - f_{min} + 1} \quad (1)$$

$$f_{max} = \max_{(u,v) \in E} \{F(u,v) | (u,v) \in E\} \quad (2)$$

$$f_{min} = \min_{(u,v) \in E} \{F(u,v) | (u,v) \in E\} \quad (3)$$

Definition 3.5 (**Footmark Graph**). *Given Ω, footmark graph $G_f = (V_f, E_f)$ is a directed sub-graph of G, and for each $u, v \in V$, $(u,v) \in E_f$ and $u, v \in V_f$ hold iff Equation 4 holds, where $w_{u,v} = F'(u,v)$ is the frequency score or weight of edge $(u,v) \in E$.*

$$0 < w_{u,v} < \frac{f_{max} + 1}{f_{max} - f_{min} + 1} \quad (4)$$

For instance, the footmark graph G_f is a directed subgraph of G w.r.t starting vertex v_1, destination vertex v_{11}, time interval $[t_s, t_e]$ and $\delta = 1.5$ as shown in Figure 3.

Definition 3.6 (**Score of Path and Most Frequent Path**). *Given Ω, G_f, the score of path $P \subseteq G_f$ is $S(P)$ as defined in Equation 5. Path $P^*_{v_s, v_d} \subseteq G_f$ is the Most Frequent Path between v_s and v_d, iff $S(P^*_{v_s, v_d}) \leq S(P_{v_s, v_d})$ holds for $\forall P_{v_s, v_d} \subseteq G_f \setminus P^*_{v_s, v_d}$.*

$$S(P) = \sum_{(u,v) \in P} w_{u,v} \quad (5)$$

Before defining TPLFP, we introduce the number of frequent enough path. Given Ω and G_f, K_p, which is also called K(or $K(P)$) value of path P, is the number of path whose normalized frequency is no larger than $\theta \overline{w}$, where $\overline{w}$ is the average weight of path in G_f, and $k_{u,v}$ is the indicator of (u,v) according to that $w_{u,v}$ is larger than $\theta \overline{w}$ or not. The detail definitions are in Equations 6, 7 and 8.

$$\overline{w} = \frac{1}{|E_f|} \sum_{(u,v) \in E_f} F'(u,v) \quad (6)$$

$$K_p = \sum_{(u,v) \in P} k_{u,v} \quad (7)$$

$$k_{u,v} = \begin{cases} 1 & W(u,v) \leq \theta \overline{w}, \\ 0 & otherwise \end{cases} \quad (8)$$

Problem Definition Given $\Omega = (G, \Upsilon, T, v_s, v_d, \delta, \varepsilon, \theta)$, $P^{\#}_{u,v}$ is the time period longest frequent path between v_s and v_d, iff Equations 9 and 10 hold at the same time. When context is clear, we use $P^{\#}$ and $P^{\#}_{u,v}$ alternatively.

$$S(P^{\#}) \leq (1+\varepsilon) S(P^*) \quad (9)$$

$$K_{P\#} = \max(K_P) \quad (10)$$

In the definition of TPLFP, parameter ε makes sure that, the frequency of $P^{\#}$ is smaller(i.e., frequent enough), while $K_{P\#}$ maximize the number of edges(i.e., road segments) whose frequency is no more than $\theta \overline{w}$, which is defined in Equation 7.

Algorithm 1: Three-stage framework of TPLFP

Input: $\Omega = (G, \Upsilon, T, v_s, v_d, \delta, \varepsilon, \theta)$
Output: the TPLFP path w.r.t Ω
constructing footmark graph G_f and finding $\overline{w}$;
find MFPs $P^*_{*v_d}$ between any vertex and v_d in G_f;
searching $P^{\#}$ w.r.t Ω, G_f and P^*;
return $P^{\#}$;

Algorithm 1 shows the three-stage framework of TPLFP. Fist, we construct footmark graph G_f w.r.t Ω online. It is a challenging work for the following two reasons: firstly, it is impossible to storage all footmark graph for all time intervals; secondly, it is impossible to storage all significant history trajectory data into memory simultaneously. Therefore, we should design a new index for big history trajectory data to construct footmark graph efficiently.

The second stage finds all the most frequent paths $P^*_{*v_d}$(including $P^*_{(v_s,v_d)}$) between other vertices and v_d through reverse Dijkstra single source shortest path algorithm. Finally, we search the TPLFP path $P^{\#}_{v_s,v_d}$ w.r.t Ω based on above two stages with some pruning strategies.

3.2. TPLFP Framework

To conduct TPLFP, we design a three-stage framework:

1) constructing footmark graph G_f between $[t_s, t_e]$;
2) finding Most Frequent Paths between to v_d;
3) searching longest frequent path between v_s and v_d.

4. Footmark Graph Construction

In footmark graph building stage, we design a filter-and-refine scheme to perform spatio-temporal querying through Linear Sketching Footmark Index, and compute edge frequency accurately.

4.1. Linear Sketch Footmark Index (LSFI)

We use Z-order curve to numerate the vertices for different time slots. As shown in Figure 4, we map the vertices to integers from 0 to 15 at the beginning. If the time slot step is every 30 minutes, their Z-order values(Z-values) become integers from 16 to 31 in the next time slot.

Supposing z^i_v is v's Z-value in ith time slot, and z^0_v is the initial value. Then, $z^i_v = z^0_v + i\max(z^0)$, where $\max(z^0)$ is the maximum Z-value of all vertices at beginning.

Through coding vertices into Z-values in different time slots, we can take Z-order integers as index item in LSFI, and one footmark will have multiple index items. We can use traditional B^+-tree to index these items, and take (Z-$order, v.id, Y.id$) as the index entry.

Figure 5 depicts the footmark index for Y'_1 to Y'_7 in Figure 2, where the index entry is $z_{y'.id}$ and the arrangement of entries is from left to right, then from top to bottom. Before performing range query on LSFI index, we should first transform the spatio-temporal range into one or more Z-order intervals, and remove the duplicate tids from results.

Algorithm 2: Footmark Graph constructing

Input: $\Omega = (G, \Upsilon, T, v_s, v_d, \delta, \varepsilon, \theta)$, index $LSFI$
Output: the footmark graph G_f and $\overline{w}$ w.r.t Ω
1 $F \leftarrow$ zero matrix of $|V(G) \times |V(G)|$;
2 Y's $\leftarrow$ **Search**$(LSFI, T, v_s, v_d, \delta)$;
3 **forall the** $Y' \in Y's$ **do**
4     $Y'_{ef} \leftarrow$ **EffectTraj**(Y', T, t_s, t_d);
5   **foreach** $(u, v) \in Y'$ **do** $F_{u,v} \leftarrow F_{u,v} + 1$;
6 $f_{max} \leftarrow$ **max**$(F_{u,v}))$, $\forall(u, v) \in Y'_{af}$;
7 $f_{min} \leftarrow$ **min**$(F_{u,v})$, $\forall(u, v) \in Y'_{af} \wedge F_{u,v} \neq 0$;
8 $F' \leftarrow$ **NormalizedFreq**(F, f_{max}, f_{min}) ;
9 $E_f \leftarrow \{(u, v)|w_{u,v} < \frac{f_{max}+1}{f_{max}-f_{min}+1}\}$;
10 $V_f \leftarrow \{u, v|(u, v) \in E_f\}$;
11 $\overline{w} \leftarrow$ **AvgWeight**$(w_{u,v})$, $\forall(u, v) \in E_f$;
12 **return** $G_f = (V_f, E_f), \overline{w}$;

Algorithm 2 describes the detailed footmark graph constructing through LSFI-index. As a running example, given input $\Omega = (G, \Upsilon, T, v_3, v_9, 1.5, \varepsilon, \theta)$, where G is defined in Figure 4, $T = [5, 30]$ is the first slot, $delta = 1.5$, therefore, line 2 retrieves trajectories through LSFI index by `search` method, in which we translate the spatio-temporal range into integer range $[0, 15]$. The retrieved trajectories is from Y'_1 to Y'_7 between t_s, t_d as shown in Figure 2 where $t_s = 5$ and $t_d = 25$. Lines 3 $\sim$ 5 compute the frequencies for each edge, where line 4 eliminates Y'_7 and points which are outside of $[t_s, t_e]$ out(in Figure 2) by method `EffectTraj`. And line 5 computes the frequency of edge (u, v) according to the number of effective footmark passing it. Next, lines 6 $\sim$ 8 normalize the frequencies F into F' according to Equation 1. At the end of Algorithm 2, lines 9 $\sim$ 11 give the edges, vertices and average weight respectively for footmark graph G_f.

Let $Y'_i(1 \leq i \leq 7)$ be a trajectory group passing the same route at the same time as described in Figure 2, the number of trajectories is $Y'_i.n$ for Y'_i group, and the sizes for all trajectory group are $(Y'_1.n, \cdots, Y'_7.n) = (3, 5, 3, 2, 1, 3, 6)$. Then, Figure 6 shows related G_f, which takes the number of passing trajectories divided by 5 as the line width.

5. Best-First Search Algorithm

In Section 4, we get the footmark graph and all most frequent paths to v_d w.r.t $\Omega = (G, \Upsilon, T, v_s, v_d, \delta, \varepsilon, \theta)$, while this section will design a **Best-First** search algorithm with some pruning strategies for TPLFP problem. Firstly, we design some strategies in Section 5.1, and then give the Best-First algorithm in Section 5.2

5.1. Pruning Strategies

Before introducing pruning strategies, we present the connective property of paths. If the destination of path P_1 is

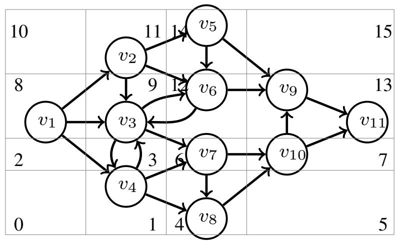


Figure 4: Z-order vertices in G

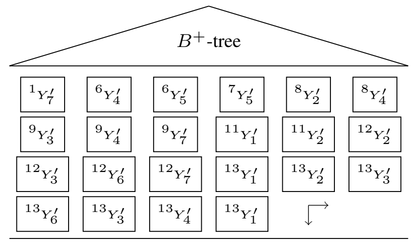


Figure 5: LSFI index

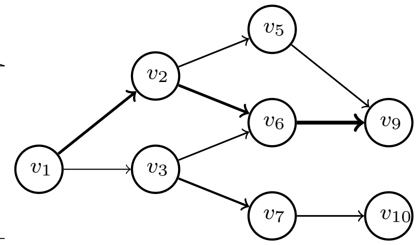


Figure 6: G_f w.r.t $G = (\Omega, \Upsilon, [5, 25], v_3, v_9, 1.5)$

the source of path P_2(i.e., $P_1.d$ equals $P_2.s$), we can connect P_2 to the end of P_1, and get a new path which takes $P_1.s$ as source and $P_2.d$ as destination. The new connected path called P_1P_2 without loop back and we call P_1 and P_2 are **connective**.

Property 5.1 (Additive property). *If path P_1 and P_2 are connective, they have the additive property described in Equation 11, in which the score of P_1P_2 is the sum of score of P_1 and P_2, and the same for the number of frequent enough edge as defined in Equations 7, 8.*

$$\begin{cases} S(P_1P_2) = S(P_1) + S(P_2) \\ K_{P_1P_2} = K_{P_1} + K_{P_2} \end{cases} \tag{11}$$

Pruning Strategy 1. *Each path $P \in P_{*v_i*v_d}$ which passing v_i can not satisfy the definition of TPLFP, if the score of sub path $P^*_{v_i*v_d}$ satisfies Equation 12. That is to say all the paths pass $v_i(v_i \neq v_s)$ to destination v_d can be pruned safely.*

$$S(P^*_{v_i*v_d}) > (1+\varepsilon)S(P^*) \tag{12}$$

Lemma 5.1 (The sub path of MFP is also MFP). *Let $P^*_{v_i,v_d}$ is the most frequent path between v_i and v_d, if $P^*_{v_i,v_d}$ passes v_m and v_n internally, then $P^*_{v_m,v_n} \in P^*_{v_i,v_d}$ is the most frequent path from v_m to v_n.*

Pruning Strategy 2. *When expanding path P_1 from v_s to v_i, if Equation 13 holds, the path P_1v_i is not an effective sub path of TPLFP , and v_i can be pruned for P_1.*

$$S(P_1) + W(P_1.d, v_i) + S(P^*_{v_i,v_d}) > (1+\varepsilon)S(P^*) \tag{13}$$

Proof. According to the notation of `Score of Path` and `Most Frequent Path` in Definition 3.6, when expanding P_1 to v_i, the score of P_1v_i becomes $S(P_1)+W(P_1.d, v_i)$ where $(P_1.d, v_i)$ is the new edge appended to P_1. However, when the remaining minimum score of path from v_i to v_d, which is the most frequent path $P^*_{v_i,v_d}$, is added to $S(P_1v_i)$ to form a complete candidate TPLFP, if the score of the complete path is large than $(1+\varepsilon)S(P^*)$, it is conflict to Equation 9, then v_i is pruned for P_1 safely. □

Definition 5.1 (Dominate). *Given paths P_1, P_2 with same source and destination, if $S(P_1) \leq S(P_2)$ and $K_{P_1} \geq K_{P_2}$, we say that P_1 dominates P_2, and describe it as $P_1 \succcurlyeq P_2$.*

Pruning Strategy 3. *Given different paths P'_1, P'_2 with same source v_s and destination v_i, if $P'_1 \succcurlyeq P'_2$, we prune P'_2 safely.*

Proof. We can make sure that $P'_1.s = P'_2.s = v_s$ and $P'_1.d = P'_2.d = v_i$ because P'_1 and P'_2 share the same source and destination. Supposing $P'' = P^{\#}_{v_i,v_d}$ is the longest frequent path between v_i and v_d, we can get $P_1 = P'_1P''$ and $P_2 = P'_2P''$ according to the connectivity of P'_1, P'_2 and P''. Therefore we can make sure that:

$$\begin{cases} S(P_1) &= S(P'_1) + S(P'') \\ S(P_2) &= S(P'_2) + S(P'') \\ K_{P_1} &= K_{P'_1} + K_{P''} \\ K_{P_2} &= K_{P'_2} + K_{P''} \end{cases}$$

Through $P'_1 \succcurlyeq P'_2$, we get $S(P'_1) < S(P'_2)$ and $K_{P'_1} > K_{P'_2}$. Then we get

$$\begin{cases} S(P'_1) + S(P'') & < & S(P'_2) + S(P'') \\ K_{P'_1} + K_{P''} & > & K_{P'_2} + K_{P''} \end{cases},$$

and combining with Equation 11 we get

$$\begin{cases} S(P_1) & < & S(P_2) \\ K_{P_1} & > & K_{P_2} \end{cases},$$

which determines that $P_1 = P_1P'' \succcurlyeq P_2 = P'_2P''$. In consequence, we can prune P_2 safely. □

Pruning Strategy 4. *Given $P'_1 = v_s \to \cdots \to v_m$, $P'_2 = v_s \to \cdots \to v_n$, P'_1 and P'_2 share none vertex except v_s, and have the relation of $P'_1 \succcurlyeq P'_2$, if $\underline{K}_{P'_1} > \overline{K}_{P'_2}$ holds, then, we can prune P'_2 safely,where $\underline{K}_{P'_1}, \overline{K}_{P'_2}$ is defined in Equation 15.*

$$\overline{S}_i = (1+\varepsilon)S(P^*) - S(P_i) \tag{14}$$

Proof. Let $P''_1 = v_m \to \cdots \to v_d$ and $P''_2 = v_n \to \cdots \to v_d$ be appended paths to P'_1 and P'_2 to form complete $v_s * v_d$ paths $P_1 = P'_1P''_1$ and $P_2 = P'_2P''_2$, both of which pass v_m and v_n respectively. In addition, $K_{P''_1}, K_{P''_2}$ are the number of paths which are frequent enough according to Equations 7 and 8.

Supposing the maximum remaining path scores which can be appended to P'_1 and P'_2 to reach destination v_d, is $\overline{S}_1$ and $\overline{S}_2$(i.e., $\overline{S}_1 = \mathbf{max}(P''_1)$ and $\overline{S}_2 = \mathbf{max}(P''_2)$) as defined in Equation 14. Then, the minimum number of frequent enough appended path for P'_1 is $\underline{K}_1 = \mathbf{min}(K(P''_1))$, and the maximum number of frequent appended enough path for P'_2 is $\overline{K}_2 = \mathbf{max}(K(P''_2))$ as defined in Equation 15.

We known that $\underline{K}_1 > \overline{K}_2$ holds as supposed in strategy 4, together with $K_{P'_1} > K_{P'_2}$(because of $P'_1 \succcurlyeq P'_2$), then we get that $K_{P'_1} + \underline{K}_1 > K_{P'_2} + \overline{K}_2$ holds, i.e., there exist a P_1'' path which has advantage in K value over every P_2'' path . Therefore, we can prune P_2' safely. $\square$

$$\begin{cases} \underline{K}_1 &= \overline{S}_1/\theta\overline{w} \\ \overline{K}_2 &= \overline{S}_2/min_{(u,v)\in E_f}(w_{u,v}) \end{cases} \tag{15}$$

5.2. Best-First Search Algorithm

Algorithm 3 describes the Best-First search algorithm. We use priority queue Q to store candidate paths which take v_s as source and forwarding to v_d as destination. First, line 1 initializes Q with a vertex MFP tuple $\{(v_s, P^*_{v_s,v_d})\}$. Each vertex v_i of path P is in company with the most frequent path from v_i to v_d, through which we can get the path score $S(P)$, $K(P), \overline{K}_P$ and $\underline{K}_p$ used in strategies easily.

Lines 2 $\sim$ 11 fetch first not-v_d-ending path P from Q, expand P with $P.d$'s neighbors to form path P', and determine whether to put them into priority queue Q w.r.t pruning strategies 1 $\sim$ 4, by which we can also remove the pruned path in Q safely. After removing the pruned paths if there exist, we should know where to put P' into Q if P' is a winner in the pruning stage. Here, we pick up the $K(\cdot)$ value as the priority criterion which determines the position where P' will exist. It is clear that path P'', which preserves higher $K(\cdot)$ score, takes precedence over others, and should be put before the smaller $S(\cdot)$ value paths; if the Ks equal between two paths, we make the path with smaller $S(\cdot)$ path value be frontier. It is clear that the path forwarding to the destination is a depth first(path with bigger $K(\cdot)$ is selected first), then greedy second(selecting smaller $S(\cdot)$ if Ks value are the same between paths) algorithm.

In details, lines 3 $\sim$ 7 fetch the first path P whose destination is not v_d. If P is not exist, the front path in Q is the longest frequent path $P^{\#}$ w.r.t Ω, and will be returned as a result in line 12. If we find path P without the v_d as destination, we expend P to its final vertex $P.d$'s neighbors to form new paths P's from line 8 to line 11, and try to add the P's into priority Q according to strategies 1 $\sim$ 4. However, if there is a path P'' in Q, which can prune P' safely, we do not add P' into Q. Otherwise, if P' prunes other P''s in Q, we add P' into Q w.r.t the priority criterion of $K(\cdot)$ and $S(\cdot)$.

Algorithm 3: Best-First Search

Input: $\Omega, G_f, P^*, \overline{w}$, revers-MFPs for v_d
Output: TPLFP w.r.t Ω

1 $Q \leftarrow \{(v_s, P^*_{v_s,v_d})\}$;
2 **while** !`Empty`(Q) **do**
3   $P =$ `Front`(Q);
4   **while** !P *and* $P.d = v_d$ **do** $P \leftarrow$ `Next`(Q, P);
5   **if** $P = \emptyset$ **then**
6     $P^{\#} \leftarrow$ `Front`(P);
7   `break`;
8   $V_{P.d} \leftarrow$ `Adjacent`$(P.d)$;
9   **foreach** $v \in V_{P.d}$ **do**
10     $P' \leftarrow P(P.d, v)$;
11     $Q \leftarrow$ `EnQueue`(Q, P') w.r.t strategies 1 $\sim$ 4;
12 `return` $P^{\#}$;

Algorithm 4: Advanced Footmark Graph Construction

Input: Ω, footmark graph G_f and average weight $\overline{w}$
Output: advanced footmark graph $G_{af} = (V_{af}, E_{af})$

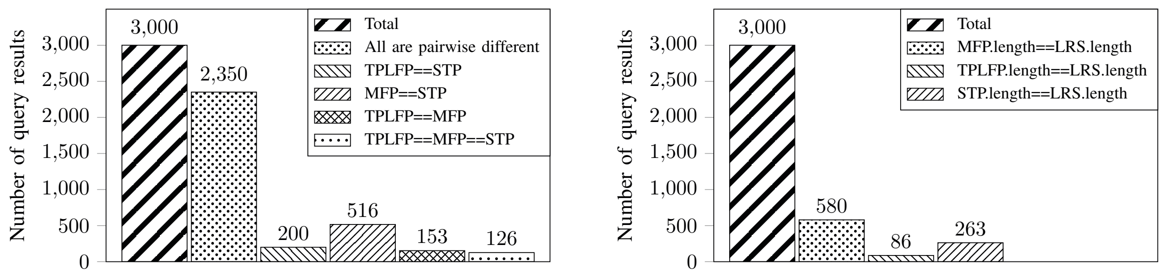
$E_{af} \leftarrow \emptyset$, $V_{af} \leftarrow \emptyset$;
foreach $e \in E_f \wedge w_e \leq \theta\overline{w}$ **do** $E_{af} \leftarrow E_{af} \cup e$;
for $\forall e_1, e_2 \in \{e | e \in E_f \wedge w_e \leq \theta\overline{w}\} \wedge e_1 \neq e_2$ **do**
  if $\exists P = P^*_{e_1.d,e_2.s} \wedge S(e_i) > \theta\overline{w}$ *for* $\forall e_i \in P$ **then**
    $E_{af} \leftarrow E_{af} \cup (e_1.d, e_2.s)$;
    $w_{e_1.d,e_2.s} \leftarrow S(P)$;
for $\forall e \in E_f \wedge w_e \leq \theta\overline{w}$ **do**
  if $\exists P = P^*_{v_s,e.s} \wedge w_{e_i} > \theta\overline{w}$ *for* $\forall e_i \in P$ **then**
    $E_{af} \leftarrow E_{af} \cup (v_s, e.s)$;
    $w_{v_s,e.s} \leftarrow S(P)$;
  if $\exists P = P^*_{e.d,v_d} \wedge w_{e_i} > \theta\overline{w}$ *for* $\forall e_i \in P$ **then**
    $E_{af} \leftarrow E_{af} \cup (e.d, v_d)$;
    $w_{e.d,v_d} \leftarrow S(P)$;
foreach $e \in E_{af}$ **do** $V_{af} \leftarrow V_{af} \cup e.s \cup e.d$;
`return` $G_{af} = (V_{af}, E_{af})$;

6. Advanced Footmark Graph

In footmark graph G_f described in Definition 3.5, there are parts of frequent enough paths, each score of which is lower than $\theta\overline{w}$. This leads numerous search branches, most of which will be pruned through one of strategies 1 $\sim$ 4 in Algorithm 3. Before being pruned, these paths expand to near the v_d mostly, and waste considerable computation.

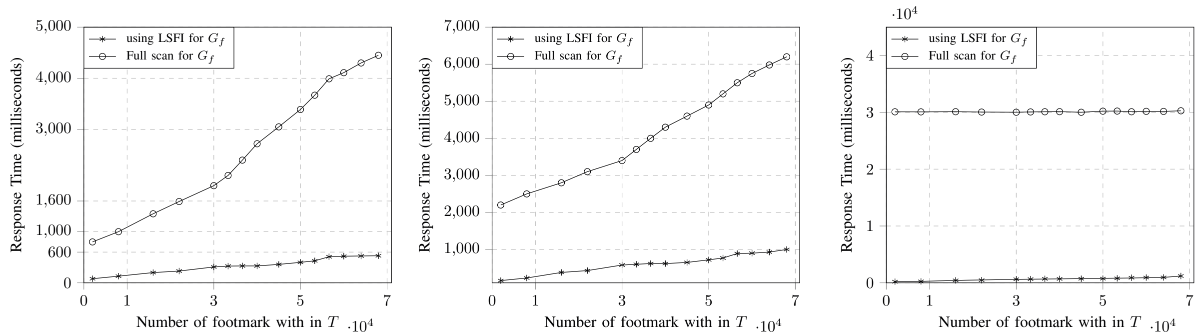
To resolve these problem, we introduce the path concentration into footmark graph, and produce an advanced footmark graph G_{af}, which is depicted in definition 6.1, and the construction method is described in Algorithm 4.

Definition 6.1 (**Advanced Footmark Graph).** $G_{af} = (V_{af}, E_{af})$ *is an advanced footmark graph w.r.t* Ω, G_f, *and has such features: 1) the* $\overline{w}$ *in* G_f *is the same with* G_f*'s; 2)* $\{e | \forall e \in E_f \wedge S(e) \leq \theta\overline{w}\} \subseteq E_{af}$ *holds, and the S(e) and* K_e *are unchanged; 3)* $e \in E_{af}$ *is a non frequent enough path iff there exist edges* $e_1, e_2 \in E_f$ *while* $e_1.d = e.s$ *and* $e.d = e_2.s$*. In addition,* $w_{e_1} \leq \theta\overline{w}$ *and* $w_{e_2} \leq \theta\overline{w}$ *hold. At the same time, there must exists most frequent path* $P^*_{e.s,e.d} \in E_f$ *which satisfies that* $w_{e_i} > \theta\overline{w}$ *holds for every* $e_i \in P^*_{e.s,e.d}$*. Finally, the weight of edge* e *equals* $S(P^*_{e.s,e.d})$*. 4) for each* $e \in E_f \wedge w_e \leq \theta\overline{w}$*, if there is a most frequent path* $P = P^*_{v_s,e.s} \in E_f$*, in which* $w_{e_i} > \theta\overline{w}$ *holds for every* $e_i \in P$*, we can make sure that there is a edge* $(v_s, e.s) \in E_{af}$*. And, it is similar to* v_d *that there is an edge* $(e.d, v_d) \in E_{af}$*. In addition,* $w_{v_s,e.s} = S(P)$ *and* $w_{e.d,v_d} = S(P^*_{e.d,v_d})$ *hold where* $P^*_{e.d,v_d} \in E_f$ *is the most frequent path between* $e.d$ *and* v_d*. 5)* $V_{af} = \{v_i, v_j | (v_i, v_j) \in E_{af}\}$*.*



(a) vs. shortest path (b) vs. least road segments

Figure 7: Statistics of results w.r.t $3,000$ pairs of source and destination vertices on Middle Dataset



(a) Small Dataset (b) Middle Dataset (c) Large Dataset

Figure 8: Footmark graph construction time using different methods

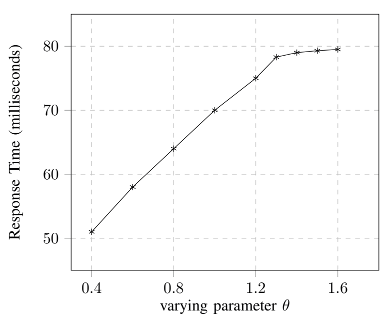


Figure 9: G_{af} construction

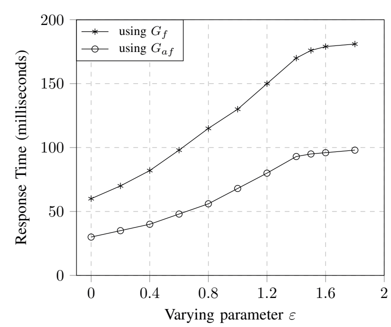


Figure 10: Best-first searching($\theta = 1$)

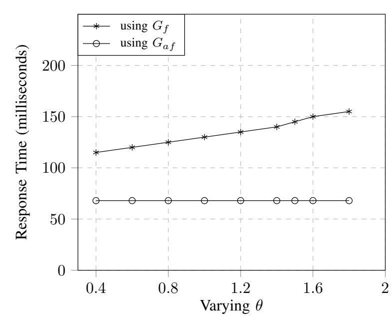


Figure 11: Best-first searching($\varepsilon = 1$)

7. Performance Evaluation

We conduct extensive experiments on a real dataset to study the performance of the proposed index and Best-First search algorithm to answering the TPLFP queries.

7.1. Experiment Settings

Dataset.We use the real dataset including over 1.3 billion GPS records from $53,136$ taxis in Beijing during Dec. 2012. After matching trajectories to Beijing's digital map($171,505$ vertices and $433,391$ edges), the average number of vertices in trajectories and Z-values are 77 and 37 respectively. For scalability test, we generate three datasets shown in Table 1.
Experiment Environment. We perform the proposed algorithms in Java on computer with a Intel i7-2600 CPU(3.4GHz), 32GB memory, $7,200$RPM, 1TB hard disk. The 64-Bit OS is Linux 3.16 and the 64-Bit JVM is 1.8.0.

TABLE 1: Summary of Datasets

Name	No. of Cars	Days	No. of Trajectories	Size
Large	$31,213$	31	$13,298,588$	48GB
Middle	$16,397$	4	$6,975,748$	25GB
Small	$3,379$	1	$1,441,399$	5GB

7.2. Effectiveness

We compare the efficiency of TPLFP with Shortest Path(STP), Least Route Segment(LRS), Most Frequent Path(MFP), and indicate the effectivity of TPLFP.

First, we focus on whether these queries find the same paths. In this setting, we generate $3,000$ pairs vertices to perform different path finding algorithms over Middle datasets with $\delta = 0.1$. The relations between TPLFP, MFP, and, STP are shown in Figure 7a. We can see that over 78% of the results are pairwise different, that is to say that each of them has unique significance and cannot be replaced by others. Also, it is important that the number of TPLFP matching STP is approximate half the number of MFP matching STP.

In Figure 7b, the number of LRSs matching MFPs is large than 19%. Clearly, they have unique significance and cannot be effectively replaced by the other. Also, the overlapped paths between TPLFP s and LRSs are less than 3%.

7.3. Efficiency

7.3.1. Effect of Footmark Graph Construction. To evaluate the efficiency of footmark graph construction, we compare Algorithm 2, Algorithm 4 and modified Algorithm 2 that performs full scan on all trajectories.

Figure 8 shows that LSFI significantly outperforms the full scan baseline approaches in all datasets. For both of them, the footmark construction time is increasing linearly with the number of trajectories. However, the construction time increases slowly through the LSFI.

In addition, Figure 9 gives the advanced footmark graph construction time when θ is changing from 0.4 to 1.6. We see that the response time is increasing when θ increases. However, when θ reaches 1.3, the $\theta\overline{w} \geq w_{u,v}$ holds for every edge (u, v), therefore G_f equals to G_{af} and the construction time is the same w.r.t θ which is no less than 1.3.

7.3.2. Effect of TPLFP queries. Next, we study the query performance with $10,000$-vertex G_f on Middle dataset when varying the parameter ε. The results are presented in Figure 10 and 11. The advanced footmark graph(G_{af}) based Best-First search algorithm provides good results than the normal footmark graph (G_f) based search.

In Figure 10, for both Best-First algorithms using G_f and G_{af}, the search time increases linearly until ε reaching 1.5. Parameter ε equaling 1.5 indicates that $S(\cdot) \leq (1 + \varepsilon)S(P^*)$ holds mostly for all the path. Then, it is clear that the search includes all edges already when ε larger than 1.5 and the search time will not increase.

Additionally, in Figure 11, the search time increase for Best-First algorithm using G_f when θ increases. However, it's nearly unchanged for Best-First algorithm using G_{af} which is not sensitive with θ. The reason is that the $w_{u,v}$ is smaller than $\theta\overline{w}$ for almost all edges in G_{af} and for all values of θ. In contrast to edges in G_{af}, the weights of edges are larger than $\theta\overline{w}$ in G_f mostly, and its number increase too when θ increases.

8. Conclusion

This paper study the problem of time period longest frequent path (TPLFP) which is a reasonable alternative most frequent path in a time period. To conduct TPLFP, we give a three-stage framework, in which, we design an index Linear Sketch Footmark Index, provide a footmark graph and its advanced version, develop a Best First search algorithm with four pruning strategies. The extensive evaluation showes that our algorithms are effective and efficient.

Acknowledgments

The work is partially supported by the National Natural Science Foundation of China (Grants No. 61402449, 91546111), Strategic and Prospective Sci&Tech Program under grant No. XDA06010600, Beijing Natural Science Foundation Key Project grant No. KZ201610005009.

References

[1] R. Bader, J. Dees, R. Geisberger, and P. Sanders. Alternative Route Graphs in Road Networks. In *Theory and Practice of Algorithms in (Computer) Systems*, pages 21–32. 2011.

[2] J. Bang-Jensen and G. Z. Gutin. *Section 2.3.4: The Bellman-Ford-Moore algorithm*. 2000.

[3] X. Cao, L. Chen, G. Cong, J. Guan, N.-T. Phan, and X. Xiao. KORS: Keyword-aware Optimal Route Search System, 2013.

[4] V. P. Chakka, A. Everspaugh, and J. M. Patel. Indexing Large Trajectory Data Sets With SETI. *CIDR*, 2003.

[5] Z. Chen, H. T. Shen, and X. Zhou. Discovering popular routes from trajectories. *ICDE*, pages 900–911, 2011.

[6] V. T. de Almeida and R. H. Guting. Indexing the trajectories of moving objects in networks, 2004.

[7] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.

[8] E. Frentzos. Indexing Objects Moving on Fixed Networks. In *Advances in Spatial and Temporal Databases*, pages 289–305. 2003.

[9] R. Geisberger, P. Sanders, D. Schultes, and D. Delling. Contraction Hierarchies: Faster and Simpler Hierarchical Routing in Road Networks. *WEA*, 5038:319–333, 2008.

[10] A. V. Goldberg, H. Kaplan, and R. F. Werneck. Reach for A*: Efficient Point-to-Point Shortest Path Algorithms, 2006.

[11] P. E. Hart, N. J. Nilsson, and B. Raphael. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Trans. on Systems Science and Cybernetics*, 4(2):100–107, 1968.

[12] H.-P. Kriegel, M. Renz, and M. Schubert. Route skyline queries: A multi-preference path planning approach, 2010.

[13] W. Luo, H. Tan, L. Chen, and L. M. Ni. Finding time period-based most frequent path in big trajectory data. *SIGMOD*, pages 713–724, 2013.

[14] D. Luxen and D. Schieferdecker. Candidate Sets for Alternative Routes in Road Networks. In *Experimental Algorithms*, pages 260–270. 2012.

[15] D. Pfoser, C. S. Jensen, and Y. Theodoridis. Novel approaches to the indexing of moving object trajectories, 2000.

[16] M. Shekelyan, G. Jossé, and M. Schubert. Linear path skylines in multicriteria networks. *ICDE*, pages 459–470, 2015.

[17] Y. Tao and D. Papadias. The MV3R-Tree: A Spatio-Temporal Access Method for Timestamp and Interval Queries. *PVLDB*, 2001.

[18] Y. Wang, Y. Zheng, and Y. Xue. Travel time estimation of a path using sparse trajectories, 2014.

[19] L.-Y. Wei, Y. Zheng, and W.-C. Peng. Constructing popular routes from uncertain trajectories. *KDD*, pages 195–203, 2012.

[20] B. Yao, M. Tang, and F. Li. Multi-approximate-keyword routing in GIS data. *GIS*, pages 201–210, 2011.

[21] C. Zhang, Y. Zhang, W. Zhang, X. Lin, M. A. Cheema, and X. Wang. Diversified Spatial Keyword Search On Road Networks. *EDBT*, pages 367–378, 2014.