Finding Time Period-Based Most Frequent Path in Big Trajectory Data[1]

Ziyang Chen

Fudan University

13307130148@fudan.edu.cn

December 23, 2014

[1] powered by XeLaTeX

Summary

Overview

- The main task: find *the most frequent*(MFP) during user-specified time periods in large-scale historical trajectory data.
- They refer to this query as *time period-based MFP*(TPMFP).
- Specifically, given a time peroid T, a source v_s and a destination v_d, TPMFP searchs the MFP from v_s to v_d during T.

Overview

- None of the previous work can well reflect people's common sense notion which can be described by the following key properties:
  - *suffix-optimal*
  - *length-insensitive*
  - *bottleneck-free*
- The first task is to give a TPMFP definition that satisfies the above three properties.
- The next task is to find TPMFP over huge amount of trajectory data efficiently.(over $11,000,000$ trajectories.)

Summary

Key Properities

Property (Suffix-Optimal)

Let P^* denote the $v_s - v_d$ MFP. For any vertex $u \in P^*$, the sub-path (suffix) of P^* from u to v_d should be the u–v_d MFP.

Property (Length-Insensitive)

The length of any path should not be a deciding factor of whether it is the $v_s - v_d$ MFP.

Property (Bottleneck-Free)

The MFP P^* should not contain infrequent edges(i.e., bottlenecks).

Summary

Definations

Defination (Road Network)

A road network is a directed graph $G = (V, E)$ where V is a set of vertices representing road intersections and E is a set of edges representing road segments.

Defination (Path)

Given G, an x_1–x_k path is a non-empty graph $P = (V_p, E_p)$ of the form $Vp = x_1, x_2, \ldots, x_k$ and $E_p = (x_1, x_2), \ldots, (x_{k-1}, x_k)$ such that P is a sub-graph of G and the x_i are all distinct.

Defination (Trajectory)

Given G, a trajectory Y is a sequence $((x_1, t_1), (x_2, t_2), \ldots, (x_k, t_k))$ such that there exists a path $x_1 \to x_2, \to, \ldots, \to x_k$ on G and t_i is a timestamp indicating the time when Y passes x_i.

Definations

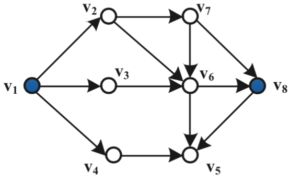
Defination (Footmark)

Given $\Omega = (G, \Upsilon, v_s, v_d, T)$ *and a trajectory* $Y = ((x_1, t_1), \ldots, (x_k, t_k)) \in \Upsilon$, *if there exists a non-empty sub-trajectory* Y' *of* Y *from* $Y[i]$ *to* $Y[j]$ *such that:*

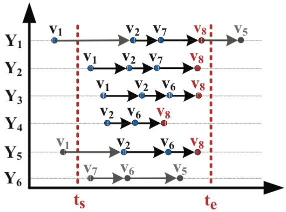
- $Y'.d = v_d, i.e., Y'[j].v = v_d,$
- $[Y'.t_s, Y'.t_e] \subseteq T, i.e., [Y[i].t, Y[j].t] \subseteq T,$
- $Y[i{-}1].t \notin T$,*if* $i > 1$,

then path $Y'.P$ *is the footmark of* Y *w.r.t.* v_d *and* T *, denoted as* $\widetilde{Y}(v_d, T)$.

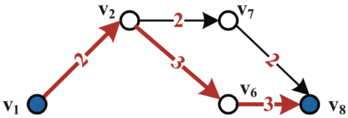
Definations



(a) Road network G



(b) Footmarks in $\widetilde{\Upsilon}$



(c) Footmark graph G_f

Definations

Defination (Edge Frequency)

Given G, $\widetilde{\Upsilon}_{(vd,T)}$, *and an edge* $(u, v) \in G$, *the edge frequency* $F(u, v)$ *is the number of the footmarks in* $\widetilde{\Upsilon}_{(vd,T)}$ *containing*(u, v).

Defination (Footmark Graph)

Given G *and* $\widetilde{\Upsilon}_{(vd,T)}$, *a footmark graph* G_f *is a weighted sub-graph of* G *such that:*

- *for any edge* $(u, v) \in G$, $w_{uv} = F(u, v)$;
- *edge* $(u, v) \in G_f$, *if and only if* $(u, v) \in G$*and* $w_{uv} > 0$.

Definations

Defination (Path Frequency)

Given G_f *, the frequency of path* $P(to\ v_d)$ *is a sequence* $F(P) = (f_1, ..., f_k)$ *where:*

- $\{f_i | i \in 1, \ldots, k\} = \{w_{uv} | (u, v) \in E(P)\}$,
- $f_1 \leq f_2 \leq \ldots \leq f_k$.

Definations

Defination (More-Frequent-Than Relation)

Given two path frequencies $F(P) = (f_1, \ldots, f_m)$ *and* $F(P') = (f'_1, \ldots, f'_n)$ *w.r.t. the same* G_f *,* $F(P)$ *is more-frequent-than* $F(P')$*, denoted as* $F(P) \succeq F(P')$*, if one of the following statements holds:*

- $F(P)$ *is a prefix of* $F(P')$*;*
- *there exists a* $q \in \{1, \ldots, min(m, n)\}$ *such that 1)* $f_i = f'_i$ *for all* $i \in \{1, \ldots, q-1\}$*, if* $q > 1$*, and 2)* $f_q > f'_q$*.*

Particularly, $F(P)$ *is strictly-more-frequent-than* $F(P')$*, denoted as* $F(P) \succ F(P')$*, if* $F(P) \succeq F(P')$ *and* $F(P) \neq F(P')$*.*

Theorem

The more-frequent-than relation is a total order.

Problem Statement

Defination (MPF)

Given G_f and a v_s–v_d path $P_* \subseteq G_f$, if $F(P_*) \succeq F(P)$ holds for every v_s–v_d path $P \subseteq G_f$, then P_* is the v_s–v_d MFP w.r.t. G_f.

Problem Statement: Given $\Omega = (G, \Upsilon, v_s, v_d, T)$ where Υ is a very large set of historical trajectories, we need to find the TPMFP which is the MFP w.r.t. G_f . Note that G_f is the footmark graph derived from Ω.

Summary

Overview Algorithm

Algorithm 1: Two major steps for the TPMFP query

Input: $\Omega = (G, \Upsilon, v_s, v_d, T)$
Output: the TPMFP w.r.t. Ω
begin
  step 1: build the footmark graph G_f w.r.t. Ω ;
  step 2: find the MFP P^* from v_s to v_d on G_f ;
  return P^* ;

Theorem

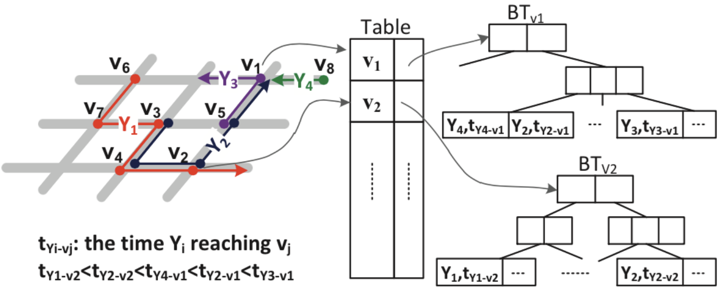
Given $\Omega = (G, \Upsilon, v_s, v_d, T)$, let P_* be the v_s–v_d TPMFP w.r.t. Ω. Then, for every vertex $u \in V(P)$, the sub-path of P_* from u to v_d is the u–v_d TPMFP.

Footmark Index

They design an index called Footmark Index (FMI):

- Build a $B^{+} - tree\ BT_{v_i}$ for each vertex $v_i \in V(G)$
- BT_{v_i} indexes the time of the trajectories reaching v_i and stores the corresponding trajectory id's
- Each leaf entry of BT_{v_i} is of the form $< tid, t_a >$
- Given v_d and T , FMI-Search(v_d, T) returns the id's of all the trajectories in $\Upsilon(v_d, T)$ via searching BT_{v_d}

Footmark Index



Footmark Index

Algorithm 2: FMI-FG(v_d, T)

begin
    $FG \leftarrow |V(G)| \times |V(G)|$ matrix with all entries zeros ;
    $TRID \leftarrow$ FMI-Search(v_d, T) ;
    for *each* $tid \in TRID$ **do**
        $Y \leftarrow$ GetTraj(tid) ;
        $(vid, t) \leftarrow$ the first element of Y ;
        while $t \notin T$ **do**
            $(vid, t) \leftarrow$ the next element of Y ;
        while $vid \neq v_d$ **do**
            $(vid', t') \leftarrow$ the next element of Y ;
            $FG[vid][vid'] \leftarrow FG[vid][vid'] + 1$;
            $(vid, t) \leftarrow (vid', t')$;
    return FG ;

Containment-Based Footmark Index

- FMI incurs $|\Upsilon(v_d, T)|$ page accesses
- Organizing the involved trajectories into different groups
- In each group, the front part of each trajectory Y before reaching v_d (including v_d), denoted as Y_{*-v_d} , is 'contained' by a unique 'dominant' trajectory
- Only need to fetch the 'dominant' trajectory
- They refer to this new index as Containment-Based Footmark Index (CFMI)

Containment-Based Footmark Index

Defination (v_d-Containment)

For two trajectories Y and Y' in Υ_{v_d} , if $Y_{*-v_d}.P$ is a sub-path of $Y'_{*-v_d}.P$, then Y is v_d-contained by Y'. In particular, if $Y_{*-v_d}.P \neq Y'_{*-v_d}.P$, then Y is stickly $v-d$-contained by Y'.

Defination (v_d-Dominant)

A trajectory $Y \in \Upsilon_{v_d}$ is v_d-dominant if there exists no $Y' \in v_d$ such that Y is strictly v_d-contained by Y'.

Containment-Based Footmark Index

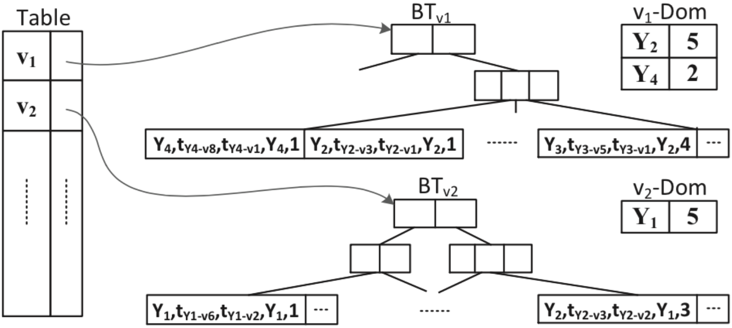
- CFMI improves the structure of each $B^+ - tree$ in FMI. Specifically, each leaf entry of BT_{v_i} is in the following new form: $< tid, t_s, t_a, did, sloc >$
- Besides, we keep a table $v_i - Dom$ for each BT_{v_i} , in which we record the length of $Y_{*-v_i}.P$ for each v_i-dominant trajectory Y

Containment-Based Footmark Index

For each query (v_i, T), CFMI returns two sets:

1. $TRREC = \{(tid, t_s, did, sloc)\}$, which records the information of trajectories in $\Upsilon(v_d, T)$
2. $DOM = \{(did, len)\}$, which records the did's appeared in TRREC and their corresponding values in $v_i - Dom$

Containment-Based Footmark Index



Containment-Based Footmark Index

Algorithm 3: CFMI-FG(v_d, T)

begin

  $FG \leftarrow |V| \times |V|$ matrix with all entries zeros ;

  $(TRREC, DOM) \leftarrow$ CFMI-Search(v_d, T) ;

  $DA \leftarrow \emptyset$;

  for *each* $(did, len) \in DOM$ **do**

    create array $DA.did[len]$ with all entries zeros ;

    $DA \leftarrow DA \cup DA.did[len]$;

  for *each* $(tid, t_s, did, sloc) \in TRREC$ **do**

    if $t_s \notin T$ **then**

      Modify-FG(tid) ;

    else

      $DA.did[sloc] \leftarrow DA.did[sloc] + 1$;

Containment-Based Footmark Index

for *each* $(did, len) \in DOM$ **do**
    $Y \leftarrow \text{GetTraj}(did)$;
    $vid \leftarrow$ the first location of $Y.P$;
    $k \leftarrow 1, w \leftarrow 0$;
    while $vid \neq v_d$ **do**
        $vid' \leftarrow$ the next location of $Y.P$;
        if $DA.did[k] \neq 0$ *or* $w \neq 0$ **then**
            $w \leftarrow w + DA.did[k]$;
            $FG[vid][vid'] \leftarrow FG[vid][vid'] + w$;
        $k \leftarrow k + 1$;
        $vid \leftarrow vid'$;
return FG ;

MFP-Search

Lemma

Let $u \rightsquigarrow v$ denote a path from u to v. Suppose $P^c = v_s \rightsquigarrow\ v_k \rightsquigarrow v_k \rightsquigarrow\ v_d$ is a path with cycles on G_f. We have $F(P) \succ F(P^c)$, where P is the resulting path after removing the portion of P^c between consecutive visits to v_k.

Lemma

Given G_f w.r.t. Ω, there exists an MFP from v_s to v_d that is simple, i.e., has at most $|V_f|-1$ edges.

Define '+' as follows:

- If the two inputs are non-decreasing sequences of positive integers, "+" merges them into a non-decreasing sequence. For example: $(20) + (5, 20) = (5, 20, 20)$;
- If one input is $\emptyset$, then the other input is returned. If both inputs are $\emptyset$'s, then $\emptyset$ is returned. For example: $\emptyset + (5, 20) = (5, 20)$;
- If one input is $\#$, then $\#$ is returned. For example: $\# + (5, 20) = \#$.

MFP-Search

Let $F^*(v_s, i)$ be the frequency of the v_s–v_d MFP using at most i edges.

Lemma

Given $G_f = (V_f, E_f)$, *if* $i > 0$, *then we have*

$$F^*(v_s, i) = \max(F^*(v_s, i–1), \max_{(v_s,v)\in E_f}((w_{v_s v}) + F^*(v, i–1))).$$

MFP-Search

Algorithm 4: MFP$(v_s, G_f = (V_f, E_f))$

begin
  for *each* $u \in V_f$ **do**
    if $u = v_d$ **then**
      $u.\xi \leftarrow \emptyset$;
    else
      $u.\xi \leftarrow \#, u.suc \leftarrow null$;
  $P^* \leftarrow null$;
  if $v_s \in V_f$ **then**
    for $i \leftarrow 1$ *to* $|V_f| - 1$ **do**
      for *each edge* $(u, v) \in E_f$ **do**
        if $(w_{uv}) + v.\xi \succeq u.\xi$ **then**
          $u.\xi \leftarrow (w_{uv}) + v.\xi$;
          $u.suc \leftarrow v$;
    create P^* by following the successors from v_s to v_d ;
  return P^* ;