Knowledge and Information Systems (2020) 62:1279–1312
https://doi.org/10.1007/s10115-019-01382-x

REGULAR PAPER

Trajectory splicing

Qiang Lu[1] · Rencai Wang[1,3] · Bin Yang[2] · Zhiguang Wang[1]

Received: 23 September 2018 / Revised: 25 June 2019 / Accepted: 30 June 2019 / Published online: 18 July 2019

Abstract

Wi h con in ed de elo men of loca ion-ba ed em , la ge amo n of ajec o ie become a ailable hich eco d mo ing objec loca ion ac o ime. If he ajec o ie collec ed b diffe en loca ion-ba ed em come f om he ame mo ing objec , he a e *spliceable trajectories*, hich con ib e o e e en ing holi ic beha io of he mo ing objec . In hi a e , e con ide ho o efficien l iden if liceable ajec o ie . Mo e ecificall , e fi fo mali e a liced model o ca e liceable ajec o ie he e hei ime a e di join , and he di ance be een hem a e clo e. Ne , o efficien l im lemen he model, e de ign h ee com onen : a di join ime inde , a di ec ed ac clic g a h of b- ajec o loca ion connec ion , and o lice algo i hm . The di join ime inde a e a di join ime e of each ajec o fo e ing di join ime ajec o ie efficien l . The di ec ed ac clic g a h con ib e o di co e ing g o of liceable ajec o ie . Ba ed on he iden ified g o , he lice algo i hm *findmaxCTR* find ma imal g o con aining all liceable ajec o ie . Al ho gh he lice algo i hm i efficien in ome ac ical a lica ion , i nning ime i e onen ial. The efo e, an a o ima e algo i hm *findApproxMaxCTR* i o o ed o find ajec o ie hich can be liced i h each o he i h a ce ain obabili i hin ol nomial n ime. Finall , e e imen on o da a e demon a e ha he model and i com onen a e effec i e and efficien .

Keywords T ajec o com a ion · T ajec o f ion · T ajec o eco e · T ajec o linking

✉ Qiang L
l iang@c .ed .cn

Rencai Wang
c ang3@ifl ek.com

Bin Yang
b ang@c .aa .dk

Zhig ang Wang
c ang g@c .ed .cn

[1] Beijing Ke Lab of Pe ole m Da a Mining, China Uni e i of Pe ole m-Beijing, Beijing, China

[2] De a men of Com e Science, Aalbo g Uni e i , Aalbo g, Denma k

[3] IFLYTEK CO.,LTD, Hefei, China

1 Introduction

Information technology is almost everywhere in our daily life, which collects various information from different digital devices [4,10]. Specially, the location-based systems based on mobile devices, such as GPS, mobile phones, and near-field communication (NFC) terminals, generate a large amount of trajectories of moving objects. Usually, each individual system uses its unique *ID* code to identify each trajectory. For example, a mobile phone network identifies a trajectory by its telephone number, while an NFC system identifies it by its device-id. Since multiple systems may capture the same moving object at different times and places, each system gathers the object's partial trajectories. Recovering a **complete trajectory** of a moving object from these partial trajectories collected in various systems, named **trajectory splicing**, is essential for many applications, such as anomaly behavior detection [21,22], data fusion, and trajectory data mining [46]. The following case shown in Fig. 1 elaborates trajectory splicing.

Every weekday, Alice and Bob go to work by walking and taking the subway, as shown in Fig. 1. Their movements generate six partial trajectories: W1, S1, O1, W2, S2, and O2, where the mobile [illegible] software captures W1 and W2; the subway check-in system captures S1 and S2; the office check-in system captures O1 and O2. Their complete trajectories can be recovered based on spatio-temporal locations of these partial trajectories. For example, S2 is more likely to splice with W2 than W1, because the endpoints' spatial positions of S2 are closer to those of W2, and the time interval of S2 [8:23,9:14] can be embedded into the time gap of W2 (8:20, 9:16). Similarly, O2 can splice with W2. So, connecting W2, S2, and O2 can recover Bob's whole trajectory.

According to the above case, finding a group of spliceable trajectories must satisfy the following three requirements. The first is the **disjoint time constraint** that requires that time intervals of spliceable trajectories in the group should not overlap with each other. The second is the **spatial constraint** that requires that the distance between their endpoints should be nearby with each other. The third is the **maximal group constraint** that requires that the group of spliceable trajectories should be maximal and should not be contained by other groups. That means connecting as many spliceable trajectories as possible to recover a complete trajectory.

However, it is non-trivial to find spliceable trajectories to satisfy the above constraints owing to the following three challenges. The first challenge is that the process of finding trajectories that satisfy the disjoint time constraint is very time-consuming. The process includes two steps: traversing sub-trajectories in all time gaps of a trajectory and counting the number of sub-trajectories that belong to the same trajectory. For example, in Fig. 1, W2 has three time gaps: $(-\infty,8:00)$, (48((c)-.107451e)15(e)-316.70003 S i

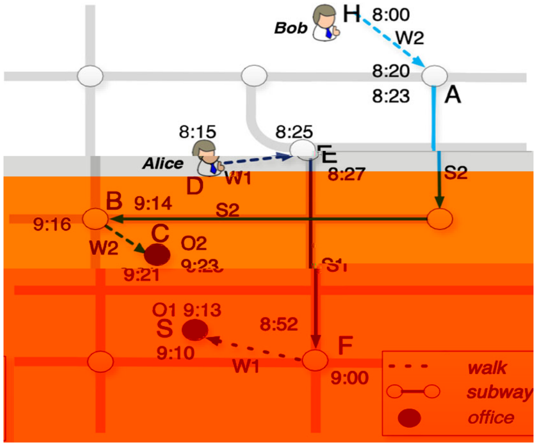


Fig. 1 The case of trajectory splicing

which connects two trajectories without going through other spliceable trajectories. The other is the **indirect splice** which connects two trajectories by going through other spliceable trajectories. For example, in Fig. 1, W2 and S2 are connected directly, while W2 and O2 are connected by S2. The indirect splice makes the process of splicing trajectories complicated because it needs to find other trajectories to determine whether the two trajectories can be connected or not. To the best of our knowledge, known group pattern mining [8,9,19,24–27,36,45] or trajectory clustering [24,25] cannot find groups of spliceable trajectories, because they discover groups of trajectories according to the similarity between them rather than the relation of direct (indirect) splice. Although future trajectory linking [38] is close to the challenge, it can only find two direct splice trajectories and is not suitable for mining multiple trajectories that are the direct or indirect splice.

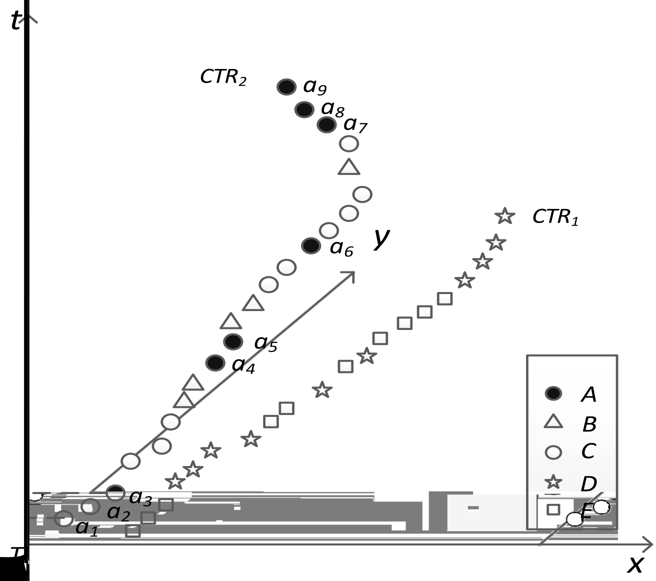
The third challenge is that it must find as many spliceable trajectories of a moving object as possible. In general, if a method wants to extract a group of spliceable trajectories which are not contained by other groups, it needs to traverse all possible combinations of spliceable trajectories for a moving object. For example, in the above case, to recover Bob's trajectory, these groups, such as {W2, S2}, {W2, O2}, {S2, O2}, and {W2, S2, O2}, must be traversed. Namely, it needs to find a group of spliceable trajectories which most compactly fill up a specific spatiotemporal range. So, it is a bin-packing problem and is NP-hard [23]. The design of an approximation scheme or heuristic method is the key to deal with the problem.

In order to deal with the above challenges, a spliced model is defined to formalize the above three requirements of spliceable trajectories. Based on the spliced model, trajectories are segmented into sub-trajectories according to a speed threshold. A B^+-tree [7] is used to save these sub-trajectories. For speeding up the process of finding disjoint time sets, the index of disjoint time called **DT-index** is constructed to keep intermediate results of searching the disjoint time sets in each time slice. Moreover, the DT-index is a multi-resolution structure like a quadtree and can save intermediate results of time slices with different lengths, supporting queries with different time intervals. For example, assuming that the DT-index consists of intermediate results of one, two, and four days, if a query time interval is 4.5 days, the DT-index can find disjoint time sets within the four days, and the B^+-tree can find disjoint time sets within the 0.5 days. Based on the above two indexes, an algorithm ***e DT TR*** is proposed to obtain all disjoint time sets within a specific time interval.

In ɑr der o find liceable ‹ ajec ɑr ie , a dir ec ed ac clic gr a h of b-‹ ajec ɑr loca ion connec ion called ***STLC-DAG*** i ɑr ea ed o connec b-‹ ajec ɑr ie b heir ime and loca ion . Once he algɑr i hm ***c ea eSTLC-DAG*** ha ɑr ea ed he gr a h, i can ob ain he liceable e of ‹ ajec ɑr ie ha can lice %i h a ecific ‹ ajec ɑr . Fɑr e am le, in he abo e ca e, he algɑr i hm can find S2 liceable e {W2}, W2 {S2, O2}, and O2 {W2}. Mɑr eo er , he e liceable e fɑr m a **splice graph**, %her e each node i a ‹ ajec ɑr , and he edge be %een %o node ‹ e ‹ e en ha he %o ‹ ajec ɑr ie ar e liceable. Fɑr in ance, he node S2 ha one edge %hich connec he node W2, and W2 ha edge %hich connec S2 and O2. Th , in he lice gr a h, a cli e i a gr o of liceable ‹ ajec ɑr ie . Fɑr addr e ing he hir d challenge, an algɑr i hm ***dMa CTR*** i ‹ o o ed o find all ma imal

Table 1 Notation

Notation	Definition
Ω	A *TR* database
p	A sample point $\langle id, lc, t\rangle$
TR_i	The i th *TR* in Ω
DT_i	A TR_i disjoint time set
SP_i	A set of *TR* that can be spliced with TR_i
STR_i^j	The j th sub-trajectory of i th *TR*
T	The time interval of set
N	The number of *TR* in T
M	The number of all *STR* in T
CTR	A complete trajectory consists of spliceable *TR*
$fst(S)$	Return the first element in sequence S
$lst(S)$	Return the last element in sequence S
$d(p, q)$	The Euclidean distance between two sample points p and q
$d(STR_i^j, STR_m^n)$	The Euclidean distance between two *STR*
$ti(STR)$	The time interval of a sub-trajectory *STR*
$ti(TR_i)$	The time interval set of trajectory TR_i
$gap(STR_i^j, STR_m^n)$	The gap between two sub-trajectories
$gap(TR_i)$	The gap set of $ti(TR_i)$



moving objects: $CTR_1 = \{TR_A, TR_B, TR_C\}$, which includes the trajectories with identifiers A, B, and C, and $CTR_2 = \{TR_D, TR_E\}$, which includes the trajectories with identifiers D and E.

In a trajectory, two sample points, p_i and p_{i+1}, are **connectable** if $speed(p_i, p_{i+1}) \geq e$, where e is a speed threshold and

$$speed(p_i, p_{i+1}) = \frac{d(p_i, p_{i+1})}{|p_{i+1}.t - p_i.t|} \tag{1}$$

where $d(p_i, p_{i+1})$ returns the Euclidean distance between two sample points p_i and p_{i+1}. Given a sequence of sample points in a trajectory TR_i, if any two consecutive sample points in the sequence are connectable, the sequence is **connectable** in that it shows one continuous movement. Moreover, if other connectable sequences do not contain a connectable sequence, the connectable sequence is called **sub-trajectory** (denoted as ***STR***). In particular, we use STR_i^j to denote the jth sub-trajectory in trajectory TR_i. For example, trajectory TR_A in Fig. 2 has 4 sub-trajectories: $STR_A^1 = \langle a_1, a_2, a_3 \rangle$, $STR_A^2 = \langle a_4, a_5 \rangle$, $STR_A^3 = \langle a_6 \rangle$, and $STR_A^4 = \langle a_7, a_8, a_9 \rangle$. A sub-trajectory is the **atomic** computational unit in this paper.

The **time interval** of the sub-trajectory, denoted as $\boldsymbol{ti}(STR)$, is $[first(STR).t, last(STR).t]$, where the functions $first(\cdot)$ and $last(\cdot)$ return the first and last sample points in the sub-trajectory STR, respectively. The **time interval** of the trajectory is the set of time intervals of all its sub-trajectories, denoted as $\boldsymbol{ti(TR_i)} = \bigcup_{STR_i^j \in TR_i} ti(STR_i^j)$.

The **gap** between two sub-trajectories STR_i^j and STR_m^n, denoted as $gap(STR_i^j, STR_m^n)$, is defined by Eq. 2.

$$gap(STR_i^j, STR_m^n) = (last(STR_i^j).t, first(STR_m^n).t) \tag{2}$$

Moreover, the **gap** of trajectory TR_i in the time interval T, denoted as $\boldsymbol{gap(TR_i)}$, is defined by Eq. 3.

$$gap(TR_i) = T - ti(TR_i) = T - \bigcup_{STR_i^j \in TR_i} ti(STR_i^j) \tag{3}$$

For example, the time interval of trajectory TR_A, denoted as $ti(TR_A)$, is $\{[t_1, t_2], [t_3, t_4], [t_5, t_5], [t_6, t_7]\}$. Given $T = [t_0, t_8]$, we have $gap(TR_A) = \{(t_0, t_1), (t_2, t_3), (t_4, t_5), (t_5, t_6), (t_7, t_8)\}$.

2.2 Spliceable trajectories

If two trajectories TR_i and TR_j can be spliced into a complete trajectory, they must meet the **disjoint time constraint** that requires that their interval times should not overlap each other, namely $ti(TR_i) \subset gap(TR_j)$. Given a trajectory TR_i, all the trajectories that meet the disjoint time constraint with TR_i constitute the **disjoint time set** of TR_i, denoted as DT_i. In Fig. 2, since $ti(TR_B) \subset gap(TR_A)$ and $ti(TR_C) \subset gap(TR_A)$, we have $DT_A = \{TR_B, TR_C\}$.

In addition to the aforementioned temporal constraints, if TR_i and TR_j are spliceable, they must also meet the **spatial constraint**, meaning that the sub-trajectories from TR_i and TR_j must be close enough to each other. To formally define the spatial constraints, we introduce two concepts: ***spliceable pair*** and ***spliceable trajectories***.

Definition 1 Given two sub-trajectories STR_i^j and STR_m^n from two trajectories, respectively, and a distance threshold γ, if they do not overlap each other on the time dimension and the distance between them is less than γ[1], the two sub-trajectories STR_i^j and STR_m^n form a ***spliceable pair***, denoted as $\langle STR_i^j, STR_m^n \rangle$.

Definition 2 Given some trajectories, if the sub-trajectories in the given trajectories can constitute a sub-trajectory sequence $\langle STR_i^j, \ldots, STR_m^n \rangle$ such that any two neighbor sub-trajectories are a spliceable pair, these trajectories are called ***spliceable trajectories***.

Based on the above two definitions, we first introduce the concept of ***complete trajectory*** to formulate the maximal group constraint, which requires that the group of spliceable trajectories should not be contained by other groups. Then, we define the ***splice degree*** to quantify the complete trajectory.

Definition 3 If other groups of spliceable trajectories do not contain a group of spliceable trajectories, the group forms a **complete trajectory**, denoted as ***CTR***.

Definition 4 The ***splice degree***, which consists of two factors: the ratio of the sum of the distance between different trajectories to the distance of *CTR* and the ratio of the sum of time gaps to the time interval of *CTR*, is used to quantify the compactness level of connection between trajectories in a *CTR*, defined by Eq. 4.

$$dg(CTR) = \frac{\sum_{\langle STR_i^j, STR_m^n \rangle \in CTR} d(STR_i^j, STR_m^n)}{distance(CTR)} \times \frac{\sum_{\langle STR_i^j, STR_m^n \rangle \in CTR} gap(STR_i^j, STR_m^n)}{time(CTR)} \tag{4}$$

where $\langle STR_i^j, STR_m^n \rangle$ is a *spliceable pair* in the *CTR*; $d(STR_i^j, STR_m^n)$ is the distance between two sub-trajectories STR_i^j and STR_m^n; $distance(CTR)$ is the sum of distance between two consecutive sample points in *CTR*, namely $distance(CTR) = \sum_{p_i \in CTR} d(p_i, p_{i+1})$, in which p_i and p_{i+1} are two consecutive sample points in the *CTR*; $time(CTR) = last(CTR).t - first(CTR).t$.

Based on the definition, $dg(CTR) \in (0, 1)$ and the smaller the splice degree $dg(CTR)$, the closer trajectories in the complete trajectory *CTR*. For example, in Fig. 2, assuming that the distance factors in Alice and Bob are the same value 0.02, $dg(\text{Alice}) = 0.02 \times (((8:27-8:25) + (9:00-8:52) + (9:13-9:10))/(9:13-8:15)) \approx 0.0448$, and $dg(\text{Bob}) = 0.02 \times ((8:23-8:20) + (9:16-9:14) + (9:23-9:21)/(9:23-8:00)) \approx 0.0017$. So, due to $dg(\text{Bob}) < dg(\text{Alice})$, the complete trajectory of Bob is better than that of Alice.

2.3 Problem definition

According to the above definitions, we formulate the problem of trajectory splicing by the trajectory splicing query.

Definition 5 From a dataset of trajectories, according to a user time interval, the ***trajectory splicing query*** discovers a complete trajectory sequence $CTRS = \langle CTR_1, \ldots, CTR_n \rangle$, where each complete trajectory *CTR* is ranked by its *splice degree*.

[1] Namely $(ti(STR_m^n) \subset gap(STR_i^j, STR_i^{j+1})) \cap (ti(STR_i^j) \subset gap(STR_m^{n-1}, STR_m^n)) \cap (d(last(STR_i^j), first(STR_m^n)) \leq \gamma)$.

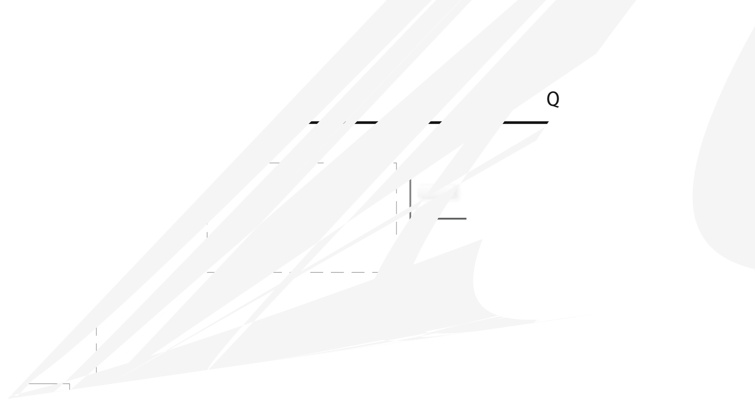

Q

Fig. 4DT-inc

where $|T| = n \times d$, d i he leng h of he ime lice, n re re en he n h ime lice, and P_i i a e which con ain all rajec orie ha a ear in T e ce he rajec or TR_i. For e am le, in Fig. 4, if $T = [0, 3d]$, $DT_D(T) = P_D^{0,3d} - [(P_D^{0,3d} - DT_D^{1,d}) \cup DF_D^{2,d} \cup DF_D^{3,d}] = \{A, B, C, E\} - [(\{A, B, C, E\} - \{E\}]) \cup \{A\} \cup \phi] = \{E\}$.

If T i oo long, here are man ime lice in T, and E . 6 con ain man nion o era ion of DF o ha he com a ion of E . 6 i ime-con ming. To alle ia e he i a ion, we ar i ion he ime dimen ion in o m l i le le el of ime lice . For in ance, one le el of ime lice i a da , and ano her le el i a week or mon h. So, if $|T|$ i one mon h, E . 6 can be com ed b onl one DF on he mon h le el of ime lice ra her han b abo 30 DF on he da le el.

(2) ***The c e f di i i e i de***

Ba ed on he abo e anal i , we de ign he di join ime inde (called ***DT***-inde) which incl de a *DT-tree* and a *DF-tree* ha a e he di join ime e DT of each rajec or and i recom a ion DF on differen le el of ime lice , re ec i el , a hown in Fig. 4. The wo ree ha e he ame r c re. The *DT-tree* (*DF-tree*) con i of a ingle roo node, leaf node , and non-roo , non-leaf node . The de ailed da a r c re of he e node are a follow .

A de, which ma ha e m l i le children, a e heir ID . A ID i bo h a ime in er al and a filename, when er ing a ime in er al T, i children and heir file are loca ed ickl .

A eaf de ore air of $\langle i, DT_i \rangle$ or $\langle i, DF_i \rangle$ in a ecific ime lice. For e am le, in Fig. 4, $DT^{3,d}$ record air $\langle A, \{B, C\} \rangle$, $\langle B, \{A, C\} \rangle$ and $\langle C, \{A, B\} \rangle$.

A - , - eaf de onl ha wo children. I ore i children ID and air of $\langle i, DT_i \rangle$ or $\langle i, DF_i \rangle$, where DT_i or DF_i can be com ed b E . 6 or 5, re ec i el .

Ai o n e 0 T D . 0 0 0 2 T c (o r) T j /

3.2 Processing query

With the B^+*-tree* and the DT-index, we implement an algorithm *QueryDTsTR* which quickly finds the disjoint time set DT of each trajectory and all sub-trajectories (denoted as *STRSet*) in a time interval T, as shown in Algorithm 1.

Algorithm 1: *queryDTsTR*

Input: B^+*-tree, DT-Index, T*
Output: *DT(T), STRSet*
1 *STRSet,* $DT(T_1), R(T_1), R(T_2)$*,* P*=readsTR*(B^+*-tree*, T) ;
2 $DT(T_2)$*= Equation 7* ;
3 $DT = (DT(T1) \cup R(T_1)) \cap (DT(T2) \cup R(T_2))$;
4 **return** *DT,STRSet* ;

The query time interval T consists of two parts: One is a set of two time intervals which are time slices in the DT-index, denoted as $T_1 = \{t_1, t_2\}$; the other is the time interval that contains n time slices in the DT-index, denoted as T_2. For example, given $T = [8:35\ 11:25]$ and the minimal time slice is an hour, $T_1 = \{[8:35\ 9:00], [11:00\ 11:25]\}$, and $T_2 = [9:00\ 11:00]$. With the B^+*-tree*, it is easy to find all trajectories P and their sub-trajectories *STRSet* in T. Meanwhile, searching these sub-trajectories can obtain a trajectory set $R(T_1)$ where each trajectory appears in T_1 but not in T_2, a trajectory set $R(T_2)$ where each trajectory appears in T_2 but not in T_1, and a disjoint time set $DT(T_1)$ in the part T_1. The function *readSTR* at Line 1 implements the above process. Then, with the DT-index, the code at Line 2 computes the disjoint time set $DT(T_2)$ by Eq. 7. At last, the code at Line 3 gets the disjoint time set DT in T.

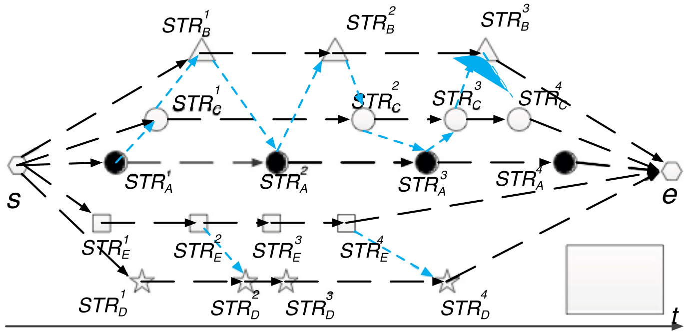
The algorithm can run very fast based on the following two reasons. One is that, in general, compared with the part T_2, the part T_1 is very short such that there are fewer sub-trajectories (*STRs*) in T_1. Hence, finding the disjoint time set $DT(T_1)$ is fast. The other is that, since the disjoint time set DT of each trajectory has been saved based on multiple time scales in the *DT*-index, only a small amount of nodes need to be searched from the index in order to compute the disjoint time set $DT(T_2)$ by Eq. 7. So, finding $DT(T_2)$ is also fast.

3.3 Splicing trajectory

3.3.1 Finding spliceable trajectories

We design an algorithm *createSTL-DAG* to discover spliceable trajectories by constructing a directed acyclic graph of sub-trajectory location connections (***STLC-DAG***), which is defined as *STLC-DAG* $= (V, E)$, where

- the vertex set V consists of all sub-trajectories (*STRSet*), a start vertex s, and an end vertex e, namely $V = \{STRSet\} \cup \{s, e\}$;
- the edge set E consists of two categories of directed edges. One, denoted as E_s, is the directed edge that connects two sub-trajectories in the same trajectory. The other, denoted as E_d, is the directed edge that connects a spliceable pair $\langle STR_i^j, STR_m^n \rangle$, as shown in Fig. 5.



STR_B^1
STR_B^2
STR_B^3
STR_C^1
STR_C^2
STR_C^3
STR_C^4
s
STR_A^1
STR_A^2
STR_A^3
STR_A^4
e
STR_E^1
STR_E^2
STR_E^3
STR_E^4
STR_D^1
STR_D^2
STR_D^3
STR_D^4
t

Since there are then multiple sequences in the graph, all first vertices from these sequences constitute a **candidate vertex set** (***CVS***), which is defined by Eq. 8.

$$CVS(STR_i^j) = \{STR_m^n | STR_m^n = first(\{ti(STR_m^k) \subset gap(STR_i^{j+1}, {}_{m}^{k}, STR_i^j)\}), m \in DT_i\} \tag{8}$$

For example, in Fig. 5, $CVS(STR_A^1) = \{STR_B^1, STR_C^1, STR_D^1, STR_E^2\}$.

Lemma 3 shows that when a trajectory cannot splice with another trajectory, the edge between the two trajectories can be deleted. Moreover, the deletion does not cause the result of spliceable trajectories to change.

The pseudocode of constructing the graph *STLC-DAG* is shown in Algorithm 2. The input arguments: the sub-trajectory set *STRSet* and the disjoint time set DT, are the result of running the algorithm *queryDTsTR*, and γ is a distance threshold. The algorithm 2 will return a set $SP = \{SP_1, \ldots, SP_n\}$, where each SP_i is a group of spliceable trajectories.

Algorithm 2: *createSTLC-DAG*

Input: $STRSet, \gamma, SP = DT$
Output: SP

1 $sortByStartTime(STRSet)$;
2 $DAG.V = STRSet \cup \{s, e\}$;
3 $DAG.E.E_s = createEsEdge(STRSet, s, e)$;
4 $C = \phi$;
5 **for** $k = 0; k < len(STRSet); k++$ **do**
6     $STR_i^j = STRSet[k]$;
7   **for** each $STR_k^v \in sortByDes(C.get(STR_i^j))$ **do**
8         $sg = 0$;
9   **repeat**
10   **if** $!existPath(STR_k^v, STR_i^j, SP_k, DAG)$ **then**
11                 $DAG.E.E_d.delEdges(TR_k, TR_i)$;
12                 $SP_i = SP_i - k$;
13                 $SP_k = SP_k - i$;
14                 $C.del(\langle TR_i, TR_m \rangle)$;
15                 $sg = |C|$;
16   **else**
17                 $sg = sg - 1$;
18             $\langle STR_k^v, STR_i^j \rangle \leftarrow C.next(STR_k^v, STR_i^j)$;
19   **until** $\langle STR_k^v, STR_i^j \rangle \neq \phi$ && $sg > 0$;
20     $canTRSet = CVS(STR_i^j)$;
21   **for** each $STR_m^n \in canTRSet$ **do**
22   **if** $d(STR_i^j, STR_m^n) \leq \gamma$ **then**
23             $DAG.E.E_d.addEdge(STR_i^j, STR_m^n)$;
24   **else**
25             $C.add(\langle STR_m^n, STR_i^j \rangle)$;
26 return SP ;

Initially, the algorithm sorts all sub-trajectories in *STRSet* by their start time, creates all vertexes, and connects the vertexes that belong to the same trajectory (Line 1-3). C is a container that saves pairs of sub-trajectories which are likely to be indirectly spliced by other sub-trajectories (Line 4). For each sub-trajectory STR_i^j in $STRSet$, its candidate vertex set $CVS(STR_i^j)$ is first obtained by Eq. 8. Then, the algorithm creates a directed edge between the two sub-trajectories STR_i^j and STR_m^n

Af e Algo i hm 2 fini he i nning, if he e e i an edge be een o ajec o ie in he g a h *STLC-DAG*, he o ajec o ie can be liced acco ding o Theo em 1. A he ame ime, he algo i hm can find g o of liceable ajec o ie SP, he e each SP_i i a e of ajec o ie ha can be di ec l o indi ec l liced i h he ajec o TR_i ba ed on Theo em 2.

Theorem 1 *If there exists a directed edge between two trajectories in the graph STLC-DAG, the two trajectories can be spliced.*

Theorem 2 *For each $SP_i \in SP$, where SP is one of the output parameters of algorithm 2, SP_i is a set of trajectories that can splice with the trajectory TR_i.*

The abo e o oof a e o ided in A endi B.

3.3.2 Finding maximum simum3ryet t5

Algorithm 5: *findApproxMaxCTR*

Input: $SP, SUBG = V, CAND = V, d, k, c = 0, fCTR = \phi$
Output: *fCTRSet*:a *fCTR* e

```
1  if SUBG! = φ then
2      if c = k then
3          if |CAND| ≤ (d − k) then
4              fCTR ← CAND;
5          else
6              fCTR ← takeFirst(CAND, d − k);
7          fCTRSet ← fCTR;
8          return ;
9      i = subscript(max|SUBG ∩ SP_i|), i ∈ SUBG;
10     branch = CAND − SP_i;
11     while branch ! = null do
12         b = takeFirst(branch) ;
13         fCTR ← b;
14         SUBG_b = SUBG ∩ SP_b;
15         CAND_b = CAND ∩ SP_b;
16         fCTRSet = findApproxMaxCTR(SP, SUBG_b, CAND_b, d, k, c + 1, fCTR);
17         CAND = CAND − {b};
18 else
19     fCTRSet ← fCTR;
20 ‹e ‹n fCTRSet;
```

Ba ed on he abo e anal i , %e de ign an algœ i hm *findApproxMaxCTR* o find a ‹o -ima e ma imal liced a h ickl . The de ailed e docode of *findApproxMaxCTR* i li ed in Algœ i hm 5. The algœ i hm i imila‹ o Algœ i hm 4 e ce he code on Line 2 8. The addi ional a‹ame e‹ a‹e a follo%: d, k, and c, %he‹e d i ed o limi he n mbe‹ of liceable ‹ajec œ ie in one com le e ‹ajec œ ; k, %hich i ed o limi he ime of in e‹ ec ion be %een %o SP , i a‹ec ‹ i e de h of he algœ i hm; and c ‹ eco‹ d he c ‹‹en ime of com ing in e‹ ec ion in a liced a h *fCTR*. The code on Line 2 8 ho% ho% o deal %i h ‹ajec œ ie in *CAND* %hen $c = k$. If he i e of *CAND* i le han $d-k$, all ‹ajec œ ie in *CAND* a‹e added in o *fCTR* (Line 3 4). If he i e i mœ e han $d-k$, he fi‹ $(d-k)$ ‹ajec œ ie a‹e added in o *fCTR* (Line 6).

4 Time complexity analysis

In hi ec ion, %e an if he‹ nning ime of he abo e algœ i hm and ignœ e algœ i hm in he ‹e‹oce ing e , ch a he con ‹ c ion of B^+-‹ee and DT-inde , beca e he can‹ n offline. Le $\boldsymbol{T}$(*function*) be he‹ nning ime of he *function*, M be he n mbe‹ of b-‹ajec œ ie , and N be he n mbe‹ of ‹ajec œ ie .

Lemma 7 *For the algorithm queryDTsTR, if the query time interval T consists of time slices from the DT-index, namely $T_1 = 0$ and $T_2 \neq 0$, the running time of queryDTsTR is $O(N^2)$; if the query time interval T does not contain the time slice for the DT-index, namely $T_2 = 0$ and $T_1 \neq 0$, the running time of queryDTsTR is $O(M^2)$.*

Proof Since all b-‹ajec œ ie a‹e inde ed b B^+-‹ee, he ime of e‹ ing m b-‹ajec œ ie i $O(log_b^{|\Omega|} + M)$. $|\Omega|$ and b a‹e con an . And, $log_b^{|\Omega|} \ll M$. So, he‹ nning time of reading all sub-trajectories in T is $O(M)$. At the same time, $R(T_1)$ and $R(T_2)$ can be obtained. If $T_1 = 0$, $DT(T_1)$ does not need to be computed. Therefore, $T(readSTR) = O(M)$. If $T_1 \neq 0$, the running time of computing $DT(T_1)$ is $O(M^2)$. And, $T(readSTR) = O(M^2)$. If $T_2 = 0$, Eq. 7 does not need to be computed. So, $T(queryDTsTR) = O(M^2)$.

If $T_2 \neq 0$, given that T_2 consists of k time slices which are in different levels in DT-index, k nodes in the DT-index need to be read. Each node contains no more than N items in which there are at most N TRs. According to Eq. 7, T(Eq. 7)$=O(kN^2)$. The running time of intersection between $DT(T_1)$ and $DT(T_2)$ is $O(N^2)$. So, *T(queryDTsTR)* is $O(N^2)$. □

Lemma 8 *The running time of the algorithm createSTLC-DAG is* $O(M^2N^2)$.

Proof Let $P = \sum_{i=1}^{N} |DT_i|$, where $DT_i \in DT$. So, $N \leq P \leq N^2$. The running time of creating vertexes (Line 3) and edges (Line 4) both are $O(M)$. In each loop (Line 5), $T(getCandSet) = O(m_k)$, where $m_k = |CVS(i, j)|$. And, the number of loops between Lines 21 and 25 also is m_k. $T(addEdge)$ and $T(add)$ both are $O(1)$. The number of creating all edges in E_d (Lines 20–25) is $\sum_{k=1}^{M} m_k$ since $len(STRSet) = M$. According to $CVS(STR_i^j)$ (Eq. 8), $m_k \leq DT_i$.

Since more sub-trajectories in TR_i result in less $|DT_i|$, the number of all edges is $\sum_{k=1}^{M} m_k$ and $\sum_{k=1}^{M} m_k \leq \frac{kM}{N} \times P$, where $k \ll N$. Moreover, running time of *pseudocode* on Lines 20–25 is $O(\frac{M}{N} \times P)$. If all edges are added into DAG (Line 23), C is empty. If all edges are added into C (Line 25), the longest time that $existPath$ runs is $\frac{M}{N} \times P$ because $delEdges$ (Line 11) can delete some edges. $T(existPath)$ depends on the number of vertexes and edges between the two sub-trajectories STR_k^v and STR_m^n. So, $T(existPath) = O(M + \frac{M}{N} \times P)$. The running time of operations on Lines 11–17 all is $O(1)$. The running time of *pseudocode* on Lines 5–19 is $O(\frac{M}{N} \times P \times (M + \frac{M}{N} \times P)) = O(\frac{M^2}{N} \times P + \frac{M^2}{N^2} \times P^2)$.

Thus, T(*createSTLC-DAG*)$= O(M + \frac{M}{N} \times P + \frac{M^2}{N} \times P + \frac{M^2}{N^2} \times P^2) = O(\frac{M^2}{N} \times P + \frac{M^2}{N^2} \times P^2) = O(\frac{M^2}{N} \times (P + \frac{P^2}{N}))$. Owing to $P \leq N^2$, T(*createSTLC-DAG*) $= O(M^2N^2)$ □

Lemma 9 *The running time of the algorithm findMaxCTR is* $O(3^{N/3})$.

Proof See Theorem 3 of [34]. □

Lemma 10 *Let D be a maximal degree of vertexes in the SP-set graph. The running time of the algorithm findApproxMaxCTR is $O(N(N-D)C_{k-1}^{D-1})$. Moreover, if k in Eq. 11 is a small numerical value, the running time of the algorithm findApproxMaxCTR is $O(CN^2)$, where C is a constant.*

Proof When the algorithm executes (depth 0) the code on Line 11 for the first time, $|branch| = N - D$. The algorithm will go to the branch SP_b, where the maximal degree of vertexes b is D. Therefore, $|SUBG_b| \leq D$. When it executes (depth 1) the code on Line 11 for the second time, $|branch| \leq D-1$. When it executes the code on Line 11 for the third time, $|branch| \leq D-2$.

Each branch recursively repeats the above process until the depth of iteration reaches k. As the depth increases, $|branch|$ decreases. Moreover, in depth $k-1$, $|branch| \leq D-k+1$. According to Theorem 1 of [34], the algorithm generates all maximal cliques with no duplication. So, each branch in the depth 1 is looked at as a combination C_{k-1}^{D-1}. The running time of $SUBG \cap SP_i$ on Line 9 is $O(N)$. Thus, T(*findApproxMaxCTR*) $= O(N(N-D)C_{k-1}^{D-1})$. When k is small, C_{k-1}^{D-1} is also small. Then, T(*findApproxMaxCTR*) $= O(CN^2)$. □

Table 2 Parameters

Notation	Definition
γ	The threshold of the distance between *STR*
d	The maximal length of a spliced path
p	Eq. 10
k	Top k complete trajectories (*CRTs*) returned by Eq. 4

5 Experiments

In this section, we present the evaluation of the trajectory splicing query (Definition 5) and its algorithms based on two large real-world trajectory datasets. The first one is Geolife [47, 48], which is used to verify the effectiveness of our algorithms because it records labeled trajectories. The other is camera trajectory, which contains trajectories generated by the road safety cameras. Moreover, camera trajectory is mainly used to test the running time of algorithms, especially the algorithm *queryDTsTR* based on the *DT*-index, because it has large amounts of trajectories.

We use the two algorithms *findMaxCTR* and *findApproxMaxCTR* to implement the trajectory splicing query, respectively. Moreover, we implement the above two algorithms in Java language on a Linux server with Intel Xeon quad-core and 8 GB of main memory. The parameters used in the following experiments are defined in Table 2.

5.1 Evaluation on geolife

5.1.1 Data set and parameter setting

In the experiments, we extract trajectories from GeoLife in 2008 as the test dataset. This test dataset contains 4405 trajectories from 32 users. Each segment of those trajectories has been labeled by one of 11 transportation modes, which are bike, boat, bus, car, run, subway, taxi, train, walk, airplane, and others. These segments are considered from 11 different datasets. So, segments from the same user with the same label make up the trajectory defined in the paper, denoted as *TR*. Each segment is the sub-trajectory defined in the paper, denoted as *STR*. The test dataset contains 138 *TRs* and 4405 *STRs*, listed in Table 3.

The function $dist(i, j)$ is the Euclidean distance between two *TRs* with two labels i and j, respectively. Table 4 lists maximum, mean, and variance of $dist(i, j)$. For example, the first row in Table 4 represents the mean, variance, and max distance between bike-*TRs* and other-*TRs*, which are 109,477 m, 146,006 m, and 212,719 m, respectively. We use four values

Table 3 Composition of *TR* Dataset

Id	Dataset	*TR*	*STR*	Id	Dataset	*TR*	*STR*
1	Airplane	1	2	7	Subway	7	108
2	Bike	14	301	8	Taxi	13	71
3	Boat	1	1	9	Train	4	12
4	Bus	22	426	10	Walk	28	756
5	Car	16	337	11	Other	30	2383
6	Run	2	8				

Table 4 Mean, Va iance and Ma in $dist(i, j)$

Dist	*Mean* (m)	*Var* (m)	*Max* (m)	*Dist*	*Mean* (m)	*Var* (m)	*Max* (m)
1, 11	109, 477	146, 006	212, 719	4, 9	133, 446	173, 046	255, 808
1, 4	14, 576	0	14, 576	5, 10	55, 642	328, 973	2, 415, 622
1, 8	293, 078	0	293, 078	5, 11	34, 362	118, 063	1, 063, 245
2, 10	1500	2777	12, 075	5, 7	8564	39, 313	267, 034
2, 11	11, 257	84, 761	1, 023, 086	5, 8	11, 348	20, 908	76, 762
2, 4	2549	3654	12, 689	5, 9	13, 957	0	13, 957
2, 5	10, 001	17, 305	52, 276	7, 10	5850	7080	31, 996
2, 7	13, 171	20, 661	44, 042	11, 7	41, 265	132, 648	637, 270
2, 8	58, 703	118, 024	269, 712	7, 8	2265	4143	11, 631
3, 4	59, 156	73	59, 207	8, 10	15, 221	26, 122	77, 098
4, 10	12, 583	84, 028	986, 741	11, 8	223, 333	1, 214, 825	8, 328, 956
4, 11	23, 340	110, 415	1, 066, 120	8, 9	761, 691	951, 360	1, 828, 952
4, 5	124, 336	548, 462	2, 517, 981	9, 10	66, 511	98, 627	235, 890
4, 6	601	1315	5516	11, 9	468, 275	466, 053	1, 245, 493
4, 7	5894	11, 273	56, 182	11, 10	20, 986	109, 772	1, 125, 060
4, 8	6966	18, 875	77, 229				

fo he a ame e γ, hich a e $\gamma = m$, $\gamma = m + v$, $\gamma = m + 1.5v$ and $\gamma = max$, he e m, v, and max a e *mean*, *var*, and *max* in Table 4, e ec i el .

5.1.2 *findMaxCTR* vs *findApproxMaxCTR*

In o de o e al a e he effec i ene of he o algo i hm ha lice ajec o ie f om he abo e 11 da a e , e define ***eca***, ***eci i***, and ***c e e e*** a E . 12, 13, and 14. *recall* e e en he abili of hich he o algo i hm can eco e com le e ajec o ie (*CTR*) f om he abo e 11 da a e ; *precision* can ho he deg ee of hich o *k CTR* con ain e ajec o ie in Geolife; *completeness* i he deg ee ha one com le e ajec o eco e a e ajec o .

$$recall = num_a / num_b \tag{12}$$

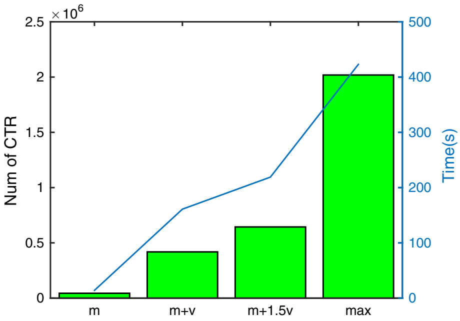
he e num_b i he n mbe of e ajec o ie in he e da a e and num_a i he n mbe of e ajec o ie fo nd b one of he o algo i hm . In hi e e imen , $num_b = 32$ d e o o al 32 e ajec o ie in he da a e .

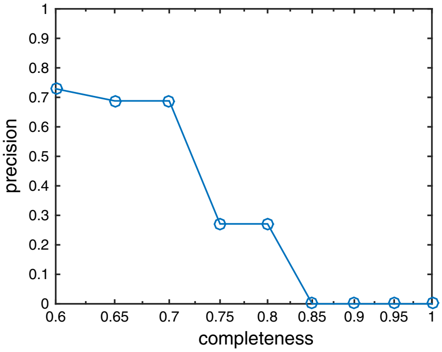
$$precision = num_c / k \tag{13}$$

he e num_c i he n mbe of com le e ajec o ie ha con ain a e ajec o ; k efe o o k com le e ajec o ie anked b E . 4.

$$completeness = \frac{|label(CTR) \cap (userTra)|}{|label(userTra)|} \tag{14}$$

he e he f nc ion $label(.)$ e n he e of an o a ion mode in a ajec o ; $|label(userTra)|$ i he n mbe of label ha a ea in a e ajec o *userTra* in he da a e ; and $|label(CTR) \cap label(userTra)|$ i he n mbe of label ha a ea bo h in *CTR* and *userTra*.


×10^6
2.5
2
1.5
1
0.5
0
Num of CTR
500
400
300
200
100
0
Time(s)
m
m+v
m+1.5v
max


precision
completeness
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0.6
0.65
0.7
0.75
0.8
0.85
0.9
0.95
1

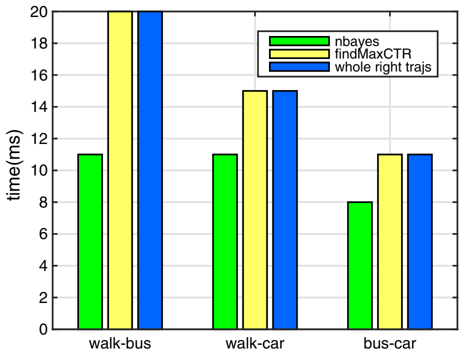


Fig. 11 *nbayes* vs *findMaxCTR* on right trajectories

5.2 Evaluation on CameraTrajectory

5.2.1 Data set and parameter setting

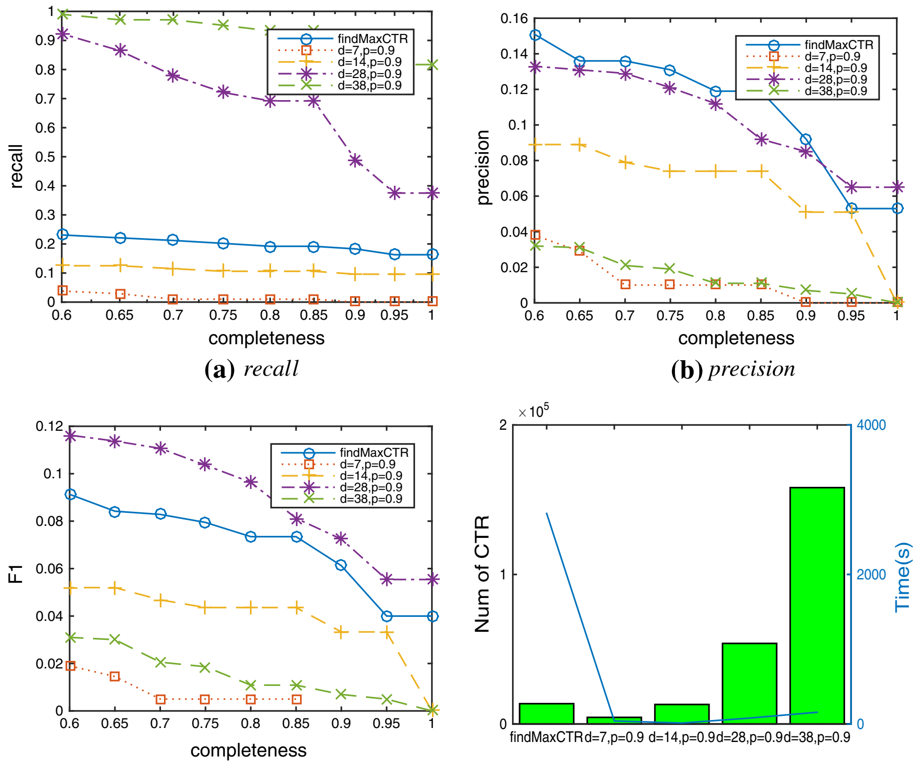
In the data set, a trajectory consists of sample points that are generated by road safety cameras, which record information of vehicles that pass by them. The data set has 10,104 trajectories and 12,741,728 sample points over three months at Guiyang, China. Since we do not know which trajectories in the data set can be spliced in advance, for computing effectiveness of the algorithms, we manually select 104 trajectories from the data set as test trajectories and randomly split these trajectories into 568 trajectories. After the two algorithms run, we observe how many complete trajectories (*CTR*s) contain these test trajectories. Thus, we can compare *recall*, *precision*, and F_1 between the two algorithms. By setting thresholds $speed = 1$ (m/s) and $distance = 10{,}000$ (m), all trajectories in the data set are split into sub-trajectories. So, there is a total of 10,568 trajectories (*TR*s) and 1,812,568 sub-trajectories (*STR*s) in the data set.

5.2.2 *findMaxCTR* vs *findApproxMaxCTR*

With the parameter $\gamma = 5000$ m, the results of *findMaxCTR* vs *findApproxMaxCTR* are shown in Fig. 12, where ($d = 7, p = 0.9$), ($d = 14, p = 0.9$), ($d = 28, p = 0.9$), and ($d = 38, p = 0.9$) are the four groups of parameters in *findApproxMaxCTR*. *findMaxCTR* finds total 13,581 groups of spliceable trajectories. However, its *recall* is about 20% as shown in Fig. 12a, because the many spliceable trajectories found by it do not satisfy the function *isSplicePath* so that they are discarded.

Compared with *findMaxCTR*, *findApproxMaxCTR* finds approximate maximal spliceable trajectories which are not checked by *isSplicePath*. Therefore, it has a higher *recall* than *findMaxCTR* when d is bigger. For example, when $d = 38$ and $p = 0.9$, its *recall*s are 82% on $completeness = 1$ and 93% on $completeness = 0.85$, respectively. However, when $d = 7$, it has a lower *recall* because the code on Line 2–8 prunes many branches that contain spliceable trajectories in Algorithm 5. So, if d is in a reasonable range, *findApproxMaxCTR* is more robust than *findMaxCTR* because its approximate results are not filtered by Definition 5.

When selecting the first 4000 results found by the two algorithms, the precisions of the two algorithms are illustrated in Fig. 12b. Compared with *findApproxMaxCTR*, *findMaxCTR* can find more user trajectories although it has a poor ability to find user trajectories with high



(a) *recall* **(b)** *precision*

(c) F1 **(d)** Max vs Approx time

Fig. 12 *findMaxCTR* er *findApproxMaxCTR*

com le ene . Acco ding o he F1 co e on Fig. 12c, *findApproxMaxCTR* %i h he fi ed a ame e ($d = 28$ and $p = 0.9$) i be e han *findMaxCTR*. Ho%e e , ea ching fo he igh a ame e al e i e o ble ome ince i need o man diffe en a ame e al e . So, f om he ie% of im lici , *findMaxCTR* i a good choice.

The ime of *findMaxCTR* nning on GeoLife (138 *TR*) i abo 160*s*, %hile i ime on Came aT ajec o (10568 *TR*) i abo 2816*s*, a ho%n in Fig . 7b and 12d. Ho%e e , i

Table 5 Components in *DT*-tree

Level	*DT*-tree		*DF*-tree	
	# of *DTNode*	*Avg size*(*kb*)	# of *DFNode*	*Avg size*(*kb*)
1	13	39,002	12	33,124
2	6	39,831	5	43,695
3	3	37,905	2	87,141

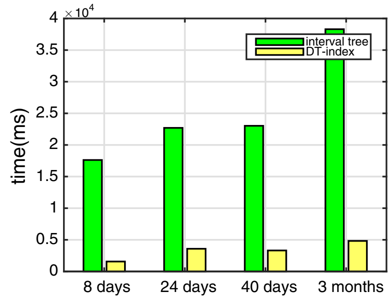


Fig. 13 B^+-tree versus *DT*-index on computing *DT*

DT-tree and the *DF*-tree both have three levels of nodes except their root nodes. The sizes of the B^+-tree and the *DT*-index are 137 Mb and 1.65 Gb, respectively, after constructing the two indexes. Table 5 lists the details of the *DT*-index. The sizes of *DTNodes* in different levels are almost the same because, according to Eq. 15, longer the time, smaller the change in the disjoint time sets of a trajectory. However, the change of sizes between *DFNodes* at different levels is big, because there is a significant difference between the two neighboring $\neg DT_i$ so that the size of DF_i is large based on $DF_i^n = \neg DT_i^n - \neg DT_i^{n-1}$. Although the size of the *DT*-index is very large, some lossless data compression algorithms, e.g., Lempel-Ziv (LZ) compression algorithm, can decrease its size. By *LZ*78 algorithm, the size of the *DT*-index changes from 1.65 Gb to 700 Mb.

As mentioned earlier in *queryDTsTR*, if $T_2 = 0$, it will search the disjoint time sets of all trajectories in the B^+-tree (called ***ITQuery***). If $T_1 = 0$, it will search all the disjoint time sets in the *DT*-index (called ***DTQuery***). After *ITQuery* and *DTQuery* run 10 times in different time intervals (8, 24, 40 days, and 3 months), their average time is shown in Fig. 13.

As a result, *DTQuery* runs faster than *ITQuery* because the time complexity of *DTQuery* is $O(N^2)$ while the time complexity of *ITQuery* is $O(M^2)$, and $M \gg N$. As the query time grows, *M* becomes bigger but *N* does not change. So, the main factor that affects the running time of *DTQuery* is only the I/O time of reading the disjoint time sets from the *DT*-index while the main

he e(<)40.099998474(,)-265.799987792( ).3(o)-706.899993896fiond <ajec or ie % ime <he di join

which is built based on the time, to store or retrieve sub-trajectories in these time intervals. Moreover, the indexes based on B^+-tree [37] and R-tree [18,33,35,40] can efficiently process the queries of time intervals. Although these indexes can process the queries, they cannot efficiently deal with the queries of time-disjoint segments because, in each query, they only support to search in a specific time interval not in multiple time intervals so that they need many queries of time intervals to discover these trajectories whose times are disjoint.

In addition to the disjoint time constraint on trajectories, spliceable trajectories require that the gap distances between them are close enough that they constitute a complete trajectory. Symbolic trajectories [13], which give a conceptual view to understand various behaviors of the moving objects [30], can capture the spliceable trajectories by a sequence of time-dependent labels. The symbolic trajectory of a moving object is represented as a sequence of units $\langle u_1, u_2, \ldots, u_n\rangle$, where u_n is a pair $\langle t, s_b, s_e, l\rangle$ in which i is a time interval, s_b and s_e are the locations of two endpoints of the unit, and l is a label. For example, for the case in Sect. 1, the symbolic trajectory of Bob is the sequence $\langle([8:00-8:20], H, A, walk), ([8:23-9:14], A, B, subway), ([9:16-9:21], B, C, walk), \ldots\rangle$.

Güting et al. [13,29,35,40] create the data model of symbolic trajectories and their indexes to offer operations to search trajectories by the above sequence of time-dependent labels. More specially, these operations support to retrieve symbolic trajectories which satisfy the conditions of the time interval, spatial distance, and a sequence of labels. For example, the retrieval SQL of Bob's transition from walk to subway is '$select\ pid\ from\ Case1\ where\ trans\ matches\ '* X(\_walk)\ Y(\_subway)*//\ Y.start - X.end \leq duration(0\ 9000000)'\ and\ pid = Bob$'. In order to match the symbolic trajectory from the database, the users must know the sequence of labels in advance. However, in the paper, the sequence of $label$ is unknown before the users begin to retrieve spliceable trajectories. So, symbolic trajectory methods do not apply to queries for the spliced mode.

Spatio-temporal joins [32,49] find close pairs of trajectories from two datasets, respectively, based on the distance between the pairs of trajectories. Based on these close pairs, the trajectory joins [1] retrieve groups of moving objects that have similar movements at a different time. Kexin Xie et al. [39] propose a spatio-temporal join method to associate segments of a trajectory with points of interest (POI) according to the distance between a POI and a trajectory and duration which a trajectory is geographically near a POI. However, the distances in these spatio-temporal join methods are the similarity between the two trajectories, while the gap distance between two trajectories is the Euclidean distance. So spatio-temporal joins are not fit to find spliceable trajectories defined in this paper because these spliceable trajectories are not similar.

6.2 Trajectory pattern analysis and mining

The spliced model needs to find groups of spliceable trajectories from different systems. Group pattern mining and trajectory clustering both find groups of moving objects based on similarity of their trajectories in a specific time interval, such as flock [8,9,36], convoy [19], swarm [27], group [26], gathering [45], and trajectory clustering methods [24,25]. These methods define different distance functions to evaluate the similarity between trajectories, and design corresponding cluster algorithms to discover groups of similar trajectories. However, these methods are not fit to find groups of spliceable trajectories because they find similar trajectories while spliceable trajectories are not similar. Another line of research on frequent trajectory mining targets at assigning travel cost-based weights to edges [15,16,42] and paths [3,44] that are frequently traversed by trajectories, where the travel cost can be travel time or fuel consumption [11,12]. However, only frequently traversed edges and paths are identified, which cannot be used directly to identify spliceable trajectories.

From the view of recovering complete user trajectories, a spliceable trajectory is one of the transportation modes in the user complete trajectory. So, discovering spliceable trajectories need to decide whether other trajectories can splice with the current trajectory based on their information about time, location, and transportation mode. Trajectory inference methods [5,28,31,46] seem to be able to make the above decision since these methods can predict a user location, infer his transportation mode, and predict when and where he will change mode [28] based on the known trajectory information. However, these methods are not good at dealing with the problem of splicing multiple trajectories owing to the two following reasons. One is that the problem of trajectory splicing acts on the different data sources while trajectory inference methods act on a single data source. In multiple data sources, each data source has a different ID code and contains trajectories of one transportation mode, and it is difficult to know in advance whether trajectories from different data sources belong to a user movement. So, the model of the problem is not built on a user history trajectory. More specifically, it is impossible to count the probability that one user switches one transportation mode to another. But, a single data source makes trajectory inference methods know user complete trajectory so that they can create their model based on user history trajectory.

The other is that they have different goals. The goal of our work is to match trajectories so that they can form one group, while the goal of trajectory inference methods is to predict a user location, infer his transportation mode, and so on. From the view of statistical learning, our work is the clustering problem, while trajectory inference methods are the regression problem. Preference learning is able to identify driver groups with similar driving preference and thus group their trajectories together [2,12,43]. However, it is unable to identify individual drivers.

The fuzzy trajectory linking(FTL) [38] is close to our work. It finds pairs of trajectories that belong to the same moving object by the two methods: (α_1, α_2)-filtering and naive Bayes matching. Compared with our methods, FTL can link (splice) two trajectories based on the distribution of distance between any two time-order points from the two trajectories, respectively. So, it avoids the disjoint time constrain in our work so that it can splice two trajectories even if their sub-trajectories overlap with each other in time. However, it does not support multiple trajectories splicing efficiently because the two above methods will be invalid as more trajectories are involved in a spliced process. Nevertheless, our methods can splice multiple trajectories. Driver identification is also similar to our work in the sense that it also tries to identify trajectories from different drivers. However, it focuses on learning distinctive representations of driving behaviors and then clusters these representations [20], but ignores disjoint time and spatial closeness.

7 Conclusion

In this paper, we study the problem of trajectory splicing, which reconstructs individual com-

For f re ork, i i of in ere o e end he liced degree b con idering o her fac or , ch a he n mber of he b-rajec orie , and he ha e of he b-rajec orie , o e al a e he ali of he recon r c ed indi id al com le e rajec or . I i al o of in ere o aral-leli e [41] he ro o ed algori hm o im ro e he efficienc and o rela he ime-di join con rain o e end he liced model o incl de more indi id al ar ial rajec orie .

Acknowledgements We o ld like o hank Profe or Chri ian S. Jen en for ef l di c ion and com-men . Thi ork a or ed b Na ional Science and Technolog Major Projec (no. 2017ZX05018-005), Na ional Na ral Science Fo nda ion of China (no. 61402532), Science Fo nda ion of China Uni er i of Pe role m-Beijing (no. 01JB0415), and China Scholar hi Co ncil.

Appendix A Computing disjoint time set

Lemma *In the query interval time T, the disjoint time set DT_i of each trajectory TR_i can be computed by Eq. 6.*

Proof Le $Q_i^{k,d}$ be a rajec or e here each rajec or TR a ear in T and i ime in er al e $ti(TR)$ doe no o erla i h TR_i)doe 11 15188-33.91659927in.0003234.600006103(R)Tj 7.1929992j -(b)989oe no im

Proof Le P_c which i fo nd b *existPath* be a a h from STR_k^v o STR_i^j. We fir 1 ro e here m e i a a h P_l from STR_k^v o STR_i^j in he c rren gra h *STLC-DAG*. P_l i an ime-orde red e ence where each $STR \in \{STR_m^n | ti(STR_k^v).st < ti(STR_m^n).st < ti(STR_i^j).st, m \in M(P_c)\} \cup \{STR_k^v, STR_i^j\}$. And, $M(P_c)$ i a e of TR ha P_c ha e a ed hro gh e ce i and k. We ro e he roblem according o he following i a ion .

If $|M(P_c)| = 0$ or $|M(P_c)| = 1$, P_c m be P_l.

If $|M(P_c)| \geq 2$, o e P_l doe no e i in he c rren *STLC-DAG*. Le P_a be he a h con ain he ma im m n mber of STR from P_l, where $M(P_c) \subseteq M(P_a)$. Then, a lea one er e STR_m^n from P_l i no on P_a. According o ime, le STR_m^n be be ween $P_a[i]$ and $P_a[i+1]$, namel $ti(P_a[i]).st < ti(STR_m^n).st < ti(P_a[i+1]).t$, where $P_a[i](P_a[j])$ i a i h or j h STR in P_a, $m_i(m_{i+1})$ i he b ori of $P_a[i](P_a[i+1])$, and $m_i, m_{i+1} \in m(P_c)$. Therefore, before r nning he c rren air, he algori hm ha e ec ed e al a ion of he wo air $\langle P_a[i], STR_m^n \rangle$ and $\langle STR_m^n, P_a[i+1] \rangle$. The e al a ion genera ed wo following re l . One i ha , if here doe no e i a a h be ween $\langle P_a[i], STR_m^n \rangle$ or $\langle STR_m^n, P_a[i+1] \rangle$, i how TR_m and $TR_{m_i}(TR_{m_{i+1}})$ canno be liced. So, $m_i \notin SP_m$ or $m_{i+1} \notin SP_m$. According o $existPath$ (Algori hm 3), i canno find ha a a h con ain $STR_{m_i}(STR_{m_{i+1}})$ and STR_m. I con radic wi h P_c. The o her i ha , if here doe e i bo h abo e a h , STR_m^n can be added in o P_a. I con radic wi h P_a ha ha he ma im m n mber of STR from P_l. Therefore, P_l m e i in he c rren *STLC-DAG*.

Then, ince P_l from STR_k^v o STR_i^j e i in *STLC-DAG*, i im lie ha here m e i a a h P_b from he ar er e o STR_k^v in he c rren *STLC-DAG*. And, P_b con ain all STR of TR be ween he ar er e and STR_k^v(P_c ha a ed hro gh he e TRs). Thi i beca e he algori hm ha roce ed re io air $\langle STR_t^r, STR_k^v \rangle$. And, here m e i a a h P_t imilar o P_a be ween STR_t^r and STR_k^v owing o he a h fo nd b *existPath*. And o on, he re io air form he P_b. Therefore, he P_b and P_l can form a liced a h. □

Lemma 5 If and onl if a a h fo nd b algori hm 3 con ain b-rajec orie from wo differen rajec orie , he wo rajec orie can be liced.

Proof If here e i a a h, which i fo nd b Algori hm 3, be ween an wo b-rajec orie from wo rajec orie , re ec i el , according o Lemma 4, he rajec orie ha he a h a ed hro gh can be liced wi h he wo b-rajec orie . So, he wo wo b-rajec orie can be liced. According o he defini ion 6, if wo b-rajec orie are liceable b-rajec orie , here e i a liced a h ha can a hro gh all b-rajec orie of he wo rajec orie . □

Theorem 1 *If there exists a directed edge between two trajectories, the two trajectories can be spliced.*

Proof S o e here i an edge be ween STR_i^j and STR_m^n, which he wo STR belong o TR_i and TR_j, re ec i el , and TR_i canno be liced wi h TR_m. According o Lemma 5, a lea one air of STR from he wo TR , re ec i el , canno be connec ed b a a h ha i fo nd b $existPath$. B , a Algori hm 2 (Line 10) m ha e dele ed all edge be ween TR_i and TR_j if i find ha a air be ween hem canno be connec ed b a a h. Therefore, here i no an edge be ween hem. I con radic he a m ion ha here i an edge be ween STR_i^j and STR_m^n.

Theorem 2 *For each $SP_i \in SP$, where SP is one of output parameters of Algorithm 2, SP_i is a set of trajectories that can be spliced with the trajectory TR_i.*

Proof A ini iali ed ha e of Algor i hm 2, $SP = DT$. S o e one SP_i ha a b r i m, and i corr e onding TR_m canno be liced wi h TR_i. Accor ding o Lemma 5, her e i no a a h be ween one air $\langle STR_i^j, STR_m^n \rangle$. And, $SP_i = SP_i - m$ (Line 12 in Algor i hm 2), ha been e ec ed. I con r adic wi h SP_i beca e SP_i con ain m. □

Lemma 6. In *SP*- e gr a h, a cli e i a gr o of liceable r ajec or ie , a ma imal cli e i a com le e r ajec or .

Proof A gr o of liceable r ajec or ie can be dir ec l or indir ec l liced wi h each o her . Ther efor e, her e e i an edge be ween an wo of hem. So, he gr o of liceable r ajec-r ie i a cli e in he gr a h. If he cli e i he ma imal cli e, he gr o of liceable r ajec or ie on he ma imal cli e canno be con ained b o her gr o . So, he ma imal cli e in he gr a h i a com le e r ajec or *CTR*. □

References

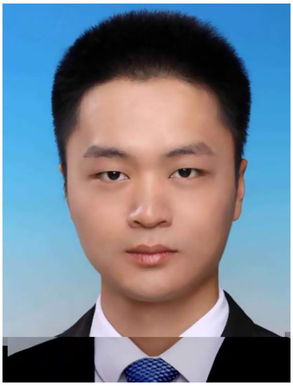
1. Bakalo P, Hadjielef her io M, T o r a VJ (2005) Time r ela ed a io em or al r ajec or join . In: Pr o-ceeding of he 13 h ann al ACM in er na ional wor k ho on geogr a hic infor ma ion em , ACM, New Yor k, NY, USA, 182 191
2. Dai J, Yang B, G o C, Ding Z (2015) Per onali ed r o e r ecommenda ion ing big r ajec or da a. In: 2015 IEEE 31 in er na ional confer ence on da a engineer ing, 543 554
3. Dai J, Yang B, G o C, Jen en CS, H J (2016) Pa h co di r ib ion e ima ion ing r ajec or da a. Pr oc VLDB Endow 10(3):85 96
4. Ding Z, Yang B, Chi Y, G o L (2016) Enabling mar r an or a ion em : a ar allel a io- em or al da aba e a r oach. IEEE Tr an Com 65(5):1377 1391
5. Emr ich T, Kr iegel HP, Mamo li N, Ren M, Z fle A (2012) Q er ing ncer ain a io- em or al da a. In: 2012 IEEE 28 h in er na ional confer ence on da a engineer ing, 354 365
6. E ein D, L ffler M, S r a h D (2010) Li ing all ma imal cli e in ar e gr a h in near -o imal ime. In: Algor i hm and com a ion, no. 6506 in lec r e no e in com er cience, S r inger Ber lin Heidelber g, 403 414
7. Goh CH, L H, Ooi BC, Tan KL (1996) Inde ing em or al da a ing e i ing B+-r ee . Da a Knowl Eng 18(2):147 165
8. G dm nd on J, an Kr e eld M (2006) Com ing longe d r a ion flock in r ajec or da a. In: Pr oceed-ing of he 14 h ann al ACM in er na ional m o i m on ad ance in geogr a hic infor ma ion em , ACM, New Yor k, NY, USA, 35 42
9. G dm nd on J, an Kr e eld M, S eckmann B (2004) Efficien de ec ion of mo ion a er n in a io-em or al da a e . In: Pr oceeding of he 12 h ann al ACM in er na ional wor k ho on geogr a hic infor ma ion em , ACM, New Yor k, NY, USA, 250 257
10. G o C, Jen en CS, Yang B (2014) Towar d o al r affic awar ene . SIGMOD Rec 43(3):18 23
11. G o C, Yang B, Ander en O, Jen en CS, Tor K (2015) Ecomar k 2.0: em ower ing eco-r o ing wi h ehic lar en ir onmen al model and ac al ehicle f el con m ion da a. GeoInfor ma ica 19(3):567 599
12. G o C, Yang B, H J, Jen en CS (2018) Lear ning o r o e wi h ar e r ajec or e . In: IEEE 34 h in er na ional confer ence on da a engineer ing, 1073 1084
13. G ing RH, Vald F, Damiani ML (2015) S mbolic r ajec or ie . ACM Tr an S a Algor i hm S 1(2):7:1 7:51
14. HCor men T, ELei er on, 99871(34 h

19. Je ng H, Yi ML, Zho X, Jen en CS, Shen HT (2008) Di co er of con o in rajec or da aba e . 1:1068 1080
20. Kie T, Yang B, G o C, Jen en CS (2018a) Di ing i hing rajec orie from differen dri er ing incom- le el labeled rajec orie . In: Proceeding of he 27 h ACM in erna ional conference on informa ion and kno%ledge managemen , 863 872
21. Kie T, Yang B, Jen en CS (2018b) O lier de ec ion for m l idimen ional ime erie ing dee ne ral ne %ork . In: IEEE 19 h in erna ional conference on mobile da a managemen , 125 134
22. Kie T, Yang B, G o C, Jen en CS (2019) O lier de ec ion for ime erie %i h rec rren a oencoder en emble . In: 28 h in erna ional join conference on ar ificial in elligence
23. Kor e B, V gen J (2012) Combina orial o imi a ion, algori hm and combina oric , ol 21. S ringer, Berlin
24. Lee JG, Han J, Whang KY (2007) Trajec or cl ering: a ar i ion-and-gro frame%ork. In: Proceeding of he 2007 ACM SIGMOD in erna ional conference on managemen of da a, ACM, Ne% York, NY, USA, 593 604
25. Lee JG, Han J, Li X (2015) A nif ing frame%ork of mining rajec or a ern of ario em oral igh ne . IEEE Tran Kno%l Da a Eng 27(6):1478 1490
26. Li X, Ceik e V, Jen en C, Tan KL (2013) Effec i e online gro di co er in rajec or da aba e . IEEE Tran Kno%l Da a Eng 25(12):2752 2766
27. Li Z, Ding B, Han J, Ka R (2010) S%arm: mining rela ed em oral mo ing objec cl er . Proc VLDB Endo% 3:723 734
28. Liao L, Pa er on DJ, Fo D, Ka H (2007) Learning and inferring ran or a ion ro ine . Ar if In ell 171(5 6):311 331
29. Sakr MA, G ing RH (2011) S a io em oral a ern erie . GeoInforma ica 15(3):497 540
30. S acca ier a S, Paren C, Damiani ML, de Macedo JA, Por o F, Vangeno C (2008) A conce al ie% on rajec orie . Da a Kno%l Eng 65(1):126 146
31. S H, Zheng K, H ang J, Wang H, Zho X (2014) Calibra ing rajec or da a for a io- em oral imilari anal i . VLDB J 24(1):93 116
32. S n J, Tao Y, Pa adia D, Kollio G (2006) S a io- em oral join elec i i . Inf S 31(8):793 813
33. Tao Y, Pa adia D (2001) MV3r-Tree: A a io- em oral acce me hod for ime am and in er al erie . In: Proceeding of he 27 h in erna ional conference on er large da a ba e , Morgan Ka fmann P bli her Inc., San Franci co, CA, USA, 431 440
34. Tomi a E, Tanaka A, Takaha hi H (2006) The %or -ca e ime com le i for genera ing all ma imal cli e and com a ional e erimen . Theor Com Sci 363(1):28 42
35. Vald F, G ing RH (2014) Inde - or ed a ern ma ching on mbolic rajec orie . In: Proceeding of he 22Nd ACM SIGSPATIAL in erna ional conference on ad ance in geogra hic informa ion em , ACM, Ne% York, NY, USA, 53 62
36. Vieira MR, Bakalo P, T o ra VJ (2009) On-line di co er of flock a ern in a io- em oral da a. In: Proceeding of he 17 h ACM SIGSPATIAL in erna ional conference on ad ance in geogra hic informa ion em , ACM, Ne% York, NY, USA, 286 295
37. Wang L, Zheng Y, Xie X, Ma WY (2008) A fle ible a io- em oral inde ing cheme for large- cale GPS rack re rie al. In: 9 h in erna ional conference on mobile da a managemen , IEEE, 1 8
38. W H, X e M, Cao J, Karra P, Ng WS, Koo KK (2016) F rajec or linking. In: IEEE 32nd in erna ional conference on da a engineering, IEEE, 859 870
39. Xie K, Deng K, Zho X (2009) From rajec orie o ac i i ie : a a io- em oral join a roach. In: Proceeding of he 2009 in erna ional %ork ho on loca ion ba ed ocial ne %ork , ACM, Ne% York, NY, USA, 25 32
40. X J, G ing RH, Zheng Y (2015) The TM-RTree: an inde on generic mo ing objec for range erie . GeoInforma ica 19(3):487 524
41. Yang B, Ma Q, Qian W, Zho A (2009) TRUSTER: rajec or da a roce ing on cl er . In: DASFAA, 768 771
42. Yang B, G o C, Jen en CS, Ka l M, Shang S (2014) S ocha ic k line ro e lanning nder ime- ar ing ncer ain . In: IEEE 30 h in erna ional conference on da a engineering, 136 147
43. Yang B, G o C, Ma Y, Jen en CS (2015) To%ard er onali ed, con e -a%are ro ing. VLDB J 24(2):297 318
44. Yang B, Dai J, G o C, Jen en CS, H J (2018) PACE: a a h-cen ric aradigm for ocha ic a h finding. VLDB J 27(2):153 178
45. Zheng K, Zheng Y, Y an N, Shang S, Zho X (2014) Online di co er of ga hering a ern o er rajec- orie . IEEE Tran Kno%l Da a Eng 26(8):1974 1988
46. Zheng Y (2015) Trajec or da a mining: an o er ie% ACM Tran In ell S Technol 6(3):1 41

47. Zheng Y, Zhang L, Xie X, Ma WY (2009) Mining interesting locations and travel sequences from GPS trajectories. In: Proceedings of the 18th international conference on World Wide Web, ACM, New York, NY, USA, pp 791–800
48. Zheng Y, Xie X, Ma WY (2010) Geolife: a collaborative social networking service among user, location and trajectory. IEEE Data Eng Bull 33(2):32–39
49. Zhou P, Zhang D, Salzberg B, Cooperman G, Kollios G (2005) Close pair queries in moving object databases. In: Proceedings of the 13th annual ACM international workshop on geographic information systems, ACM, New York, NY, USA, pp 2–11

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Qiang Lu received a B.S. degree from Shenyang University of Chemical Technology, Shenyang, China, in 2000 and a Ph.D. degree from China University of Petroleum-Beijing, China, 2006. From 2015 to 2016, he was a visiting scholar at the Department of Computer Science, Aalborg University, Denmark. He is currently an Associate Professor in the Department of Computer Science and Director of Computation Intelligence Center at China University of Petroleum, Beijing. He is also a faculty member in Beijing Key Lab of Petroleum Data Mining. His research interests include spatial-temporal data processing, evolutionary computing, and machine learning.



Rencai Wang received a B.S. degree from China University of Petroleum-East China, in 2014 and an M.S. degree from China University of Petroleum-Beijing, China, in 2017. He is currently working as a software engineer at IFLYTEK CO., LTD, responsible for data analysis and mining on education, and the development of the education cloud platform. His research interests include spatial temporal data management, trajectory computing, and data mining on behavior of the educational user.

Bin Yang i a Pr ofe or in he De ar men of Com er Science a Aalbor g Uni er i , Denmar k. He %a a Aar h Uni er i , Denmar k and a Ma Planck In i e for Infor ma ic , Ger man . He r ecei ed he Ph.D. degr ee in com er cience fr om F dan Uni er i . Hi r e ear ch in er e incl de machine lear ning and da a managemen . He %a a PC co-chair of IEEE MDM 2018. He ha er ed on r ogr am commi ee and a an in i ed r e ie%er for e er al in er na ional confer ence and jo r nal , incl ding ICDE, IJCAI, TKDE, he VLDB Jo r nal, and ACM Com ing S r e .

Zhiguang Wang r ecei ed a B.S. degr ee in h ic fr om Inner Mongolia Nor mal Uni er i in 1986, an M.S. degr ee in com er me er ing fr om Jilin Uni er i in 1994, and a Ph.D. degr ee in com er cience fr om China Uni er i of Pe r ole m-Beijing. He i c r r en l a Pr ofe or in he De ar men of Com er Science a China Uni er i of Pe r ole m, Beijing, e er ed a Dir ec or of Re ear ch Gr o of Lar ge Scale Da a Pr oce ing and Vi ali a ion. He i al o a fac l member in Beijing Ke Lab of Pe r ole m Da a Mining, and a co ncil member in Beijing Ed ca ion Feder a ion. Hi c r r en r e ear ch in er e incl de da a managemen , di r ib ed em, and a ial- em or al da a mining.