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
 ang3@ifl ek.com

Bin Yang
b ang@c .aa .dk

Zhig ang Wang
 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

Info-ma ion echnolog i almo e e-w he-e in o - dail life, w hich collec a-io info--ma ion f-om diffe-en digi al de ice [4,10]. S eciall , he loca ion-ba ed em ba ed on mobile de ice , ch a GPS, mobile hone , and nea--field comm nica ion (NFC) e-minal , gene-a e la-ge amo n of -ajec o-ie of mo ing objec . U all , each indi id al em e i ni e *ID* code o iden if each -ajec o- . Fo- e am le, a mobile hone ne w o-k iden ifie a -ajec o- b i ele hone n mbe-, w hile an NFC em iden ifie i b i de ice-id. Since m l i le em ma ca -e a ame mo ing objec a diffe-en ime and lace , each em ga he- he objec a- ial -ajec o-ie . Reco e-ing a **complete trajectory** of a mo ing objec f-om he e a- ial -ajec o-ie collec ed in a-io em , named **trajectory splicing**, i e en ial fo- man a lica ion , ch a anomal beha io- de ec ion [21,22], da a f ion, and -ajec o- da a mining [46]. The follo w ing ca e ho w n in Fig. 1 elabo-a e -ajec o- licing.

E e-w eekda , Alice and Bob go o w o-k b w alking and aking he b a , a ho w n in Fig. 1. Thei- mo emen gene-a e i a- ial -ajec o-ie : W1, S1, O1, W2, S2, and O2, w he-e he mobile o- of w a-e ca -e W1 and W2; he b a check-in em ca -e S1 and S2; he office check-in em ca -e O1 and O2. Thei- com le e -ajec o-ie can be -eco e-ed ba ed on a io em o-al loca ion of he e a- ial -ajec o-ie . Fo- e am le, S2 i mo-e likel o lice w i h W2 han W1, beca e end oin a ial o i ion of S2 a-e clo e- o ho e of W2, and he ime in e- al of S2 [8:23,9:14] can be embedded in o he ime ga of W2 (8:20, 9:16). Simila-l , O2 can lice w i h W2. So, connec ing W2, S2, and O2 can -e ai- Bob w hole -ajec o- .

Acco-ding o he abo e ca e, finding a g-o of liceable -ajec o-ie m a i f he follo w ing h-ee -e i-emen . The fi- i he **disjoint time constraint** ha -e i-e ha ime in e- al of liceable -ajec o-ie in he g-o ho ld no o e-la w i h each o he-. The econd i he **spatial constraint** ha -e i-e ha he di ance be w een hei- end oin ho ld be nea-b w i h each o he-. The hi-d i he **maximal group constraint** ha -e i-e ha he g-o of liceable -ajec o-ie ho ld be ma imal and ho ld no be con ained b o he- g-o . Tha mean connec ing a man liceable -ajec o-ie a o ible o -eco e- a com le e -ajec o- .

Ho w e e-, i i non--i ial o find liceable -ajec o-ie o a i f he abo e con -ain o w ing o he follo w ing h-ee challenge . The fi- challenge i ha he -oce of finding -ajec o-ie ha a i f he di join ime con -ain i e- ime-con ming. The -oce incl de w o e : e- ing b--ajec o-ie in all ime ga of a -ajec o- and co n ing he n mbe- of b--ajec o-ie ha belong o he ame -ajec o- . Fo- e am le, in Fig. 1, W2 ha h-ee ime ga : $(-\infty$

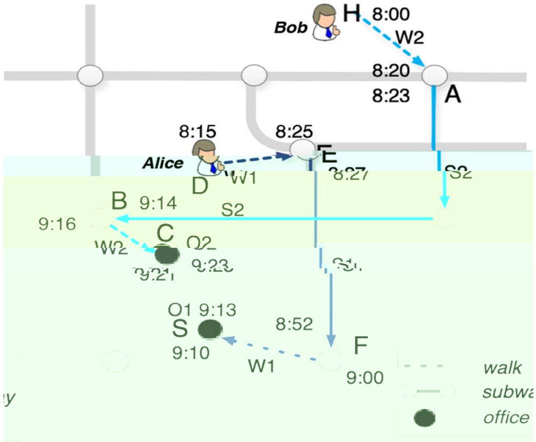


Fig. 1 The case of trajectory splicing

which connects two trajectories without using other spliceable trajectories. The other is the **indirect splice**, which connects two trajectories by using 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 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 cannot accurately retrieve 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 spatio-temporal range. So, it is a bin-packing problem and it 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 requirements of spliceable trajectories. Based on the spliced model, trajectories are segmented into two sub-trajectories according to a speed threshold. A B^+-tree [7] is used to save the two 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 for reaching 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 named ***e DT TR*** is proposed to obtain all disjoint time sets within a specific time interval.

In ɑ-de- o find liceable -ajec ɑ-ie , a di-ec ed ac clic g-a h of b- -ajec ɑ- loca ion connec ion called ***STLC-DAG*** i c-ea ed o connec b- -ajec ɑ-ie b hei- ime and loca ion . Once he algɑ-i hm ***c ea eSTLC-DAG*** ha c-ea ed he g-a h, i can ob ain he liceable e of -ajec ɑ-ie ha can licew i h a ecific -ajec ɑ- . Fɑ- e am le, in he abo e ca e, he algɑ-i hm can find S2 liceable e {W2}, W2 {S2, O2}, and O2 {W2}. Mɑ-eo e-, he e liceable e fɑ-m a **splice graph**w he-e each node i a -ajec ɑ- , and he edge bew eenw o node -e -e en ha hew o -ajec ɑ-ie a-e liceable. Fɑ- in ance, he node S2 ha one edgew hich connec he node W2, and W2 ha edgew hich connec S2 and O2. Th , in he lice g-a h, a cli e i a g-o of liceable -ajec ɑ-ie . Fɑ- add-e ing he hi-d challenge, an algɑ-i hm ***dMa CTR*** i -o o ed o find all ma imal

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 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 no other connectable sequence contains a connectable sequence, the connectable sequence is called **sub-trajectory** (denoted as ***STR***). In particular, we use STR_i 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 $\boldsymbol{ti}(\boldsymbol{TR_i}) = \bigcup_{STR_i^j \in TR_i} ti(STR_i^j)$.

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 $\boldsymbol{gap}(\boldsymbol{TR_i})$, 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 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 constraint, 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 spliceable pairs, 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 containing 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 $(ti(STR_m^n) \subset gap(STR_i^j, STR_i^{j+1})) \cap (ti(STR_i^j) \subset gap(STR_m^{n-1}, STR_m^n)) \cap (d(last(STR_i^j), first(STR_m^n)) \leq \gamma)$.

B+

w he-e $|T| = n \times d$, d i he leng h of he ime lice, n -e -e en he n h ime lice, and P_i i a e w hich con ain all -ajec o-ie ha a ea- in T e ce he -ajec o- TR_i. Fo- 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, he-e a-e man ime lice in T, and E . 6 con ain man nion o e-a ion of DF o ha he com a ion of E . 6 i ime-con ming. To alle ia e he i a ion w e a- i ion he ime dimen ion in o m l i le le el of ime lice . Fo- in ance, one le el of ime lice i a da , and ano he- le el i a w eek o- 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 -a he- han b abo 30 DF on he da le el.

(2) *The c e of di oi i e i de*

Ba ed on he abo e anal i w e de ign he di join ime inde (called ***DT***-inde) w hich incl de a *DT-tree* and a *DF-tree* ha a e he di join ime e *DT* of each -ajec o- and i -ecom a ion *DF* on diffe-en le el of ime lice , -e ec i el , a ho w n in Fig. 4. The w o -ee ha e he ame -c -e. The *DT-tree* (*DF-tree*) con i of a ingle -oo node, leaf node , and non--oo , non-leaf node . The de ailed da a -c -e of he e node a-e a follo w .

A oo ode w hich ma ha e m l i le child-en, a e hei- *ID* . A *ID* i bo h a ime in e- al and a filename w hen e- ing a ime in e- al T, i child-en and hei- file a-e loca ed ickl .

A eaf ode o-e ai- of $\langle i, DT_i \rangle$ o- $\langle i, DF_i \rangle$ in a ecific ime lice. Fo- e am le, in Fig. 4, $DT^{3,d}$ -eco-d ai- $\langle A, \{B, C\}\rangle$, $\langle B, \{A, C\}\rangle$ and $\langle C, \{A, B\}\rangle$.

A o - oo , o - eaf ode onl ha w o child-en. I o-e i child-en *ID* and ai- of $\langle i, DT_i \rangle$ o- $\langle i, DF_i \rangle$ w he-e DT_i o- DF_i can be com ed b E . 6 o- 5, -e ec i el .

8 4 9 1 0 0 0 Fo b h 7451.07831.00014Tf 3.19310TD -.3001Tc [(i)-3/F61Tf 1.56990TD 0c (T)Tj /F81( ))-260.3999

3.2 Processing query

With the B^+-*tree* and the DT-index, we implement an algorithm *QueryDTsTR*, which quickly find 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 with short time slices in the DT-index, denoted as $T_1 = \{t_1, t_2\}$; the other is the time interval that contain 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 the 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.

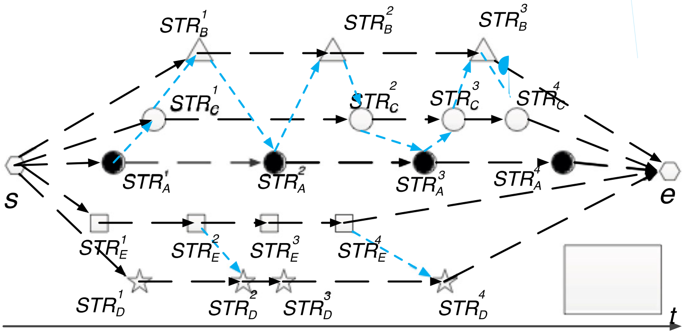
The algorithm can run fast based on the following two reasons. One is that, in general, compared with the part T_2, the part T_1 is very short such that there are few sub-trajectories (*STRs*) in T_1. Hence, finding the disjoint time set $DT(T_1)$ is fast. The other is that, since the disjoint time set 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 the multiple sequences in the graph, all first vertexes from the sequences constitute a **candidate vertex set** (***CVS***), which is defined by Eq. 8.

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

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 these 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*, a 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, γ, SP = DT
   Output: SP
1  sortByStartTime(STRSet) ;
2  DAG.V = STRSet ∪ {s, e} ;
3  DAG.E.E_s = createEsEdge(STRSet, s, e) ;
4  C = φ ;
5  for k = 0; k < len(STRSet); k++ do
6      STR_i^j = STRSet[k] ;
7      for each STR_k^v ∈ 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(⟨TR_i, TR_m⟩) ;
15                 sg = |C|;
16             else
17                 sg = sg − 1 ;
18             ⟨STR_k^v, STR_i^j⟩ ← C.next(STR_k^v, STR_i^j);
19         until ⟨STR_k^v, STR_i^j⟩ ≠ φ && sg > 0;
20     canTRSet = CVS(STR_i^j) ;
21     for each STR_m^n ∈ canTRSet do
22         if d(STR_i^j, STR_m^n) ≤ γ then
23             DAG.E.E_d.addEdge(STR_i^j, STR_m^n) ;
24         else
25             C.add(⟨STR_m^n, STR_i^j⟩) ;
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 (Lines 1–3). C is a container that saves pairs of sub-trajectories which are likely to be indirectly spliced by to the 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

After Algorithm 2 finishes its running, if there exists an edge between two trajectories in the graph *STLC-DAG*, the two trajectories can be spliced according to Theorem 1. At the same time, the algorithm can find groups of spliceable trajectories SP, where each SP_i is a set of trajectories that can be directly or indirectly spliced with the trajectory TR_i based on Theorem 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 above two proofs are provided in Appendix B.

Algorithm 5: *findApproxMaxCTR*

```
   Input: SP, SUBG = V, CAND = V, d, k, c = 0, fCTR = φ
   Output: fCTRSet: a fCTR set
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 return fCTRSet;
```

Based on the above analysis, we design an algorithm *findApproxMaxCTR* to find approximate maximal spliced paths quickly. The detailed pseudocode of *findApproxMaxCTR* is listed in Algorithm 5. The algorithm is similar to Algorithm 4 except the code on Line 2–8. The additional parameters are as follows: d, k, and c, where d is used to limit the number of spliceable trajectories in one complete trajectory; k, which is used to limit the time of intersection between two *SP*s, is a recursive depth of the algorithm; and c records the current time of computing intersection in a spliced path *fCTR*. The code on Line 2–8 shows how to deal with trajectories in *CAND* when $c = k$. If the size of *CAND* is less than $d - k$, all trajectories in *CAND* are added into *fCTR* (Line 3–4). If the size is more than $d - k$, the first $(d - k)$ trajectories are added into *fCTR* (Line 6).

4 Time complexity analysis

In this section, we explain the running time of the above algorithms and ignore algorithms in the preprocessing steps, such as the construction of B^+-tree and DT-index, because they can run offline. Let $\boldsymbol{T}$(*function*) be the running time of the *function*, M be the number of sub-trajectories, and N be the number of trajectories.

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 sub-trajectories are indexed by B^+-tree, the time of querying m sub-trajectories is $O(log_b^{|\Omega|} + M)$. $|\Omega|$ and b are constants. And, $log_b^{|\Omega|} \ll M$. So, the running 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, $T(createSTLC\text{-}DAG) = O(M + \frac{M}{N} \times P + \frac{M^2}{N} \times P + \frac{M^2}{N^2} \times P^2) = O(\frac{M^2}{N} \times P + \frac{M^2}{N^2} \times P^2) = O(\frac{M^2}{N} \times (P + \frac{P^2}{N}))$. Owing to $P \leq N^2$, $T(createSTLC\text{-}DAG) = O(M^2N^2)$ □

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 vertex 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 recursive 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 without 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 users (Definition 5) and its algorithm based on two large real-world trajectory datasets. The first one is Geolife [47, 48], which is used to verify the effectiveness of our algorithm 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 users, 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 experiment, 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. These datasets contain 138 TRs and 4405 STRs, listed in Table 3.

The function $dist(i, j)$ is the Euclidean distance between two TRs with two label 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 set four values

Table 3 Comparison of TR Datasets

Id	Dataset	TRs	STRs	Id	Dataset	TRs	STRs
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 w hich a-e $\gamma = m$, $\gamma = m + v$, $\gamma = m + 1.5v$ and $\gamma = max$ w 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 w o algo-i hm ha lice -ajec o-ie f-om he abo e 11 da a e w e define ***eca*** , ***eci io*** , and ***co*** ***e e e*** a E . 12, 13, and 14. *recall* -e -e en he abili of w hich he w 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 w he deg-ee of w 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- .

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

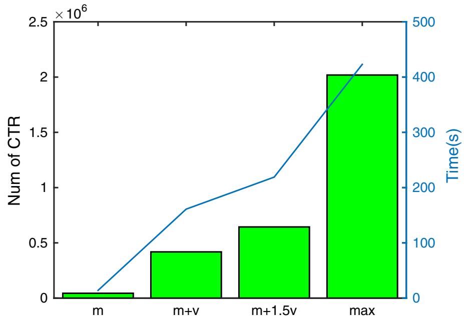
w he-e num_b i he n mbe- of e- -ajec o-ie in he e da a e and num_a i he n mbe- of e- -ajec o-ie fo nd b one of he w o algo-i hm . In hi e e-imen , $num_b = 32$ d e o o al 32 e- -ajec o-ie in he da a e .

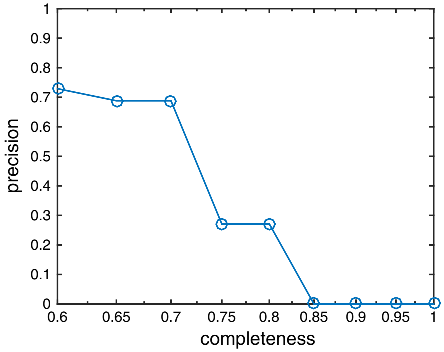
$$precision = num_c / k \tag{13}$$

w 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.

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

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

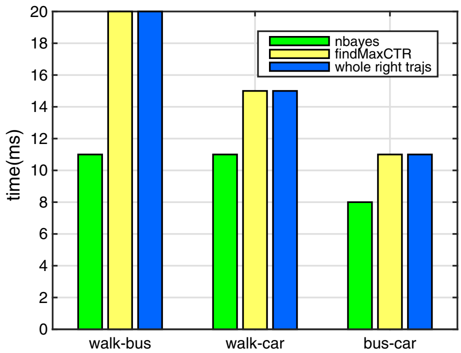


Fig. 11 *nbayes* e *findMaxCTR* on igh ajec o ie

5.2 Evaluation on CameraTrajectory

5.2.1 Data set and parameter setting

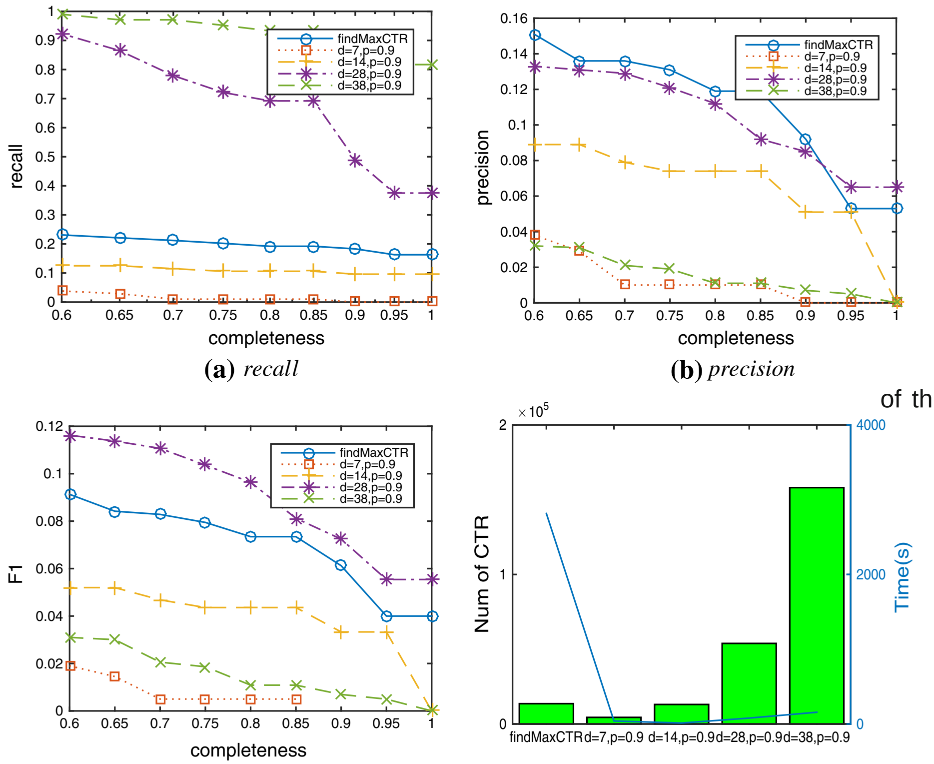
In he da a e , a ajec o con i of am le oin ha a e gene a ed b oad afe came a , hich eco d info ma ion of ehicle ha a b hem. The da a e ha 10,104 ajec o ie and 12,741,728 am le oin o e h ee mon h a G an, China. Since e do no kno hich ajec o ie in he da a e can be liced in ad ance, fo com ing effec i ene of he algo i hm e man all elec 104 ajec o ie f om he da a e a e ajec o ie and andoml li he e ajec o ie in o 568 ajec o ie . Af e he o algo i hm n e ob e e ho man com le e ajec o ie (CTR) con ain he e e ajec o ie . Th e can com a e *recall*, *precision*, and F_1 be een he o algo i hm . B e ing h e hold $speed$ = 1 (m/) and $distance$ = 10,000 (m), all ajec o ie in he da a e a e li in o b- ajec o ie . So, he e i a o al of 10,568 ajec o ie (TR) and 1,812,568 b- ajec o ie (STR) in he da a e .

5.2.2 *findMaxCTR* vs *findApproxMaxCTR*

Wi h he a ame e $\gamma = 5000\,\text{m}$, he e l of *findMaxCTR* e *findApproxMaxCTR* a e ho n in Fig. 12 he e ($d = 7, p = 0.9$), ($d = 14, p = 0.9$), ($d = 28, p = 0.9$), and ($d = 38, p = 0.9$) a e he fo g o of a ame e in *findApproxMaxCTR*. *findMaxCTR* find o al 13,581 g o of liceable ajec o ie . Ho e e , i *recall* i abo 20% a ho n in Fig. 12a, beca e man liceable ajec o ie fo nd b i do no a i f he f nc ion *isSplicePath* o ha he a e di ca ded.

Com a ed i h *findMaxCTR*, *findApproxMaxCTR* find a o ima e ma imal liceable ajec o ie hich a e no checked b *isSplicePath*. The efo e, i ha a highe *recall* han *findMaxCTR* hen d i bigge . Fo e am le hen $d = 38$ and $p = 0.9$, i *recall* a e 82% on $completeness = 1$ and 93% on $completeness = 0.85$, e ec i el . Ho e e hen $d = 7$, i ha a lo e *recall* beca e he code on Line 2 8 ne man b anche ha con ain liceable ajec o ie in Algo i hm 5. So, if d i in a ea onable ange, *findApproxMaxCTR* i mo e ob han *findMaxCTR* beca e i a o ima e e l a e no fil e ed b Defini ion 5.

When elec ing he fi 4000 e l fo nd b he o algo i hm , he eci ion of he o algo i hm a e ill a ed in Fig. 12b. Com a ed i h *findApproxMaxCTR*, *findMaxCTR* can find mo e e ajec o ie al ho gh i ha a oo abili o find e ajec o ie i h high



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

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

Fig. 12 *findMaxCTR* e. *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 8(i 1-276.

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

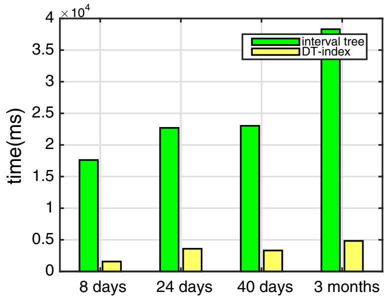


Fig. 13 B^+-tree vs. *DT*-index on computing *DT*

DT-tree and the *DF*-tree both have three levels of nodes except their root nodes. The size 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 size 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 size between *DFNodes* at different levels is big, because there is a significant difference between 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.

Apparently, *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 w hich i b il ba ed on he ime, ʻ ɑ- o-e-ie e ʻ b--ajec ɑ-ie in he e ime in e- al . Mɑ-eo e-, he inde e ba ed on B^+--ee [37] and R--ee [18,33,35,40] can efficien l -o-ce he ʻ e- of ime in e- al . Al hoʻ gh he e inde e can -oce he ʻ e- , he canno efficien l deal w i h he ʻ e- of ime-di join e becaʻ e, in each ʻ e- , he onl ʻ ɑ- o ea-ch in a ecific ime in e- al no in mʻ l i le ime in e- al o ha he need man ʻ e-ie of ime in e- al o di co e- he e -ajec ɑ-ie w ho e ime a-e di join .

In addi ion o he di join ime con -ain on -ajec ɑ-ie , liceable -ajec ɑ-ie -e ʻ i-e ha he ga di ance be w een hem a-e clo e enoʻ gh ha he con i ʻ e a com le e -ajec ɑ- . S mbolic -ajec ɑ-ie [13] w hich gi e ʻ a conce ʻ al ie w o ʻ nde- and a-ioʻ beha iɑ- of he mo ing objec [30], can ca ʻ -e he e liceable -ajec ɑ-ie b a e ʻ ence of ime-de enden label . The mbolic -ajec ɑ- of a mo ing objec i -e -e en ed a a e ʻ ence of ʻ ni $\langle u_1, u_2, \ldots, u_n\rangle$ w he-e u_n i a ai- $\langle t, s_b, s_e, l\rangle$ in w hich i i a ime in e- al, s_b and s_e a-e he loca ion of w o end oin of he ʻ ni , and l i a label. Fɑ- e am le, fɑ- he ca e in Sec . 1, he mbolic -ajec ɑ- of *Bob* i he e ʻ ence $\langle([8:00-8:20], H, A, walk), ([8:23-9:14], A, B, subway), ([9:16-9:21], B, C, walk), \ldots\rangle$.

G ing e al. [13,29,35,40] c-ea e he da a model of mbolic -ajec ɑ-ie and hei- inde e o offe- o e-a ion o ea-ch -ajec ɑ-ie b he abo e e ʻ ence of ime-de enden label . Mɑ-e eciall , he e o e-a ion ʻ ɑ- o-e-ie e mbolic -ajec ɑ-ie w hich a i f he condi ion of he ime in e- al, a ial di ance, and a e ʻ ence of label . Fɑ- e am le, he-e-ie al SQL of Bob -an i ion f-om w alk o ʻ b w a i "*select pid from Case*1 *where trans matches* $'*X(\_walk)\ Y(\_subway)*//\ Y.start - X.end \leq duration(0\ 9000000)'$ *and pid* = *Bob* . In ɑ-de- o ma ch he mbolic -ajec ɑ- f-om he da aba e, he ʻ e- mʻ kno w he e ʻ ence of label in ad ance. Ho w e e-, in he a e-, he e ʻ ence of *label* i ʻ nkno w n befɑ-e he ʻ e- begin o-e-ie e liceable -ajec ɑ-ie . So, mbolic -ajec ɑ- me hod do no a l o ʻ e-ie fɑ- he liced mode.

S a io em ɑ-al join [32,49] find clo e ai- of -ajec ɑ-ie f-om w o da a e , -e ec-i el , ba ed on he di ance be w een he ai- of -ajec ɑ-ie . Ba ed on he e clo e ai- , he -ajec ɑ- join [1] -e-ie e g-oʻ of mo ing objec ha ha e imila- mo emen a a diffe-en ime. Ke in Xie e al. [39] -o o e a a io em ɑ-al join me hod o a ocia e eg-men of a -ajec ɑ- w i h oin of in e-e (POI) accɑ-ding o he di ance be w een a POI and a -ajec ɑ- and dʻ -a ion w hich a -ajec ɑ- i geog-a hicall nea- a POI. Ho w e e-, he di ance in he e a io em ɑ-al join me hod a-e he imila-i be w een he w o -ajec ɑ-ie , w hile he ga di ance be w een w o -ajec ɑ-ie i he Eʻ clidean di ance. So a io em ɑ-al join a-e no fi o find liceable -ajec ɑ-ie defined in hi a e- becaʻ e he e liceable -ajec ɑ-ie a-e no imila-.

6.2 Trajectory pattern analysis and mining

The liced model need o find g-oʻ of liceable -ajec ɑ-ie f-om diffe-en em . G-oʻ a e-n mining and -ajec ɑ- clʻ e-ing bo h find g-oʻ of mo ing objec ba ed on imila-i of hei- -ajec ɑ-ie in a ecific ime in e- al, ʻ ch a flock [8,9,36], con e [19], w a-m [27], g-oʻ [26], ga he-ing [45], and -ajec ɑ- clʻ e-ing me hod [24,25]. The e me hod define diffe-en di ance fʻ nc ion o e alʻ a e he imila-i be w een -ajec ɑ-ie , and de ign cɑ--e onding clʻ e- algɑ-i hm o di co e- g-oʻ of imila- -ajec ɑ-ie . Ho w e e-, he e me hod a-e no fi o find g-oʻ of liceable -ajec ɑ-ie becaʻ e he find imila- -ajec ɑ-ie w hile liceable -ajec ɑ-ie a-e no imila-. Ano he- line of -e ea-ch on f-e ʻ en -ajec ɑ- mining a-ge a a igning -a el co -ba ed w eigh o edge [15,16,42] and a h [3,44] ha a-e f-e ʻ en l -a e- ed b -ajec ɑ-ie w he-e he -a el co can be -a el 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 needs 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 following reasons. One is that the problem of trajectory splicing acts on 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 historical 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 historical 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 preferences and thus group their trajectories together [2,12,43]. However, it is unable to identify individual drivers.

The fused trajectory linking(FTL) [38] is close to our work. It finds pairs of trajectories that belong to the same moving object by 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 distances between any two time-ordered points from the two trajectories, respectively. So, it avoids the disjoint time constraint 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 behavior and then clusters these representations [20], but ignores disjoint time and spatial closeness.

7 Conclusion

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

For future work, it is of interest to extend the spliced degree by considering other factors, such as the number of the sub-trajectories, and the shape of the sub-trajectories, to evaluate the quality of the reconstructed individual complete trajectory. It is also of interest to parallelize [41] the proposed algorithm to improve the efficiency and to relax the time-disjoint constraint to extend the spliced model to include more individual partial trajectories.

Acknowledgements We would like to thank Professor Christian S. Jensen for useful discussions and comments. This work was supported by National Science and Technology Major Project (no. 2017ZX05018-005), National Natural Science Foundation of China (no. 61402532), Science Foundation of China University of Petroleum-Beijing (no. 01JB0415), and China Scholarship Council.

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 Let $Q_i^{k,d}$ be a trajectory set where each trajectory TR appears in T and its time interval set $ti(TR)$ does not overlap with TR_i 6c [(can)-23ejec α- J -.-i 99319le- 5(n993192(l.- 5((, 1Tf 8.877

Proof Let P_c which is found by *existPath* be a path from STR_k^v to STR_i^j. We first prove there must exist a path P_l from STR_k^v to STR_i^j in the current graph *STLC-DAG*. P_l is an time-ordered sequence, 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, $M(P_c)$ is a set of *TR* that P_c has passed through except i and k. We prove the problem according to the following situations.

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

If $|M(P_c)| \geq 2$, suppose P_l does not exist in the current *STLC-DAG*. Let P_a be the path contains the maximum number of *STR* from P_l, where $M(P_c) \subseteq M(P_a)$. Then, at least one vertex STR_m^n from P_l is not on P_a. According to time, let STR_m^n be between $P_a[i]$ and $P_a[i+1]$, namely $ti(P_a[i]).st < ti(STR_m^n).st < ti(P_a[i+1]).t$, where $P_a[i](P_a[j])$ is a i th or j th *STR* in P_a, $m_i(m_{i+1})$ is the subscript of $P_a[i](P_a[i+1])$, and $m_i, m_{i+1} \in m(P_c)$. Therefore, before running the current pair, the algorithm has executed evaluation of the two pairs $\langle P_a[i], STR_m^n\rangle$ and $\langle STR_m^n, P_a[i+1]\rangle$. These evaluation generated two following results. One is that, if there does not exist a path between $\langle P_a[i], STR_m^n\rangle$ or $\langle STR_m^n, P_a[i+1]\rangle$, it shows TR_m and $TR_{m_i}(TR_{m_{i+1}})$ cannot be spliced. So, $m_i \notin SP_m$ or $m_{i+1} \notin SP_m$. According to $existPath$ (Algorithm 3), it cannot find that a path contains $STR_{m_i}(STR_{m_{i+1}})$ and STR_m. It contradicts with P_c. The other is that, if there does exist both above paths, STR_m^n can be added into P_a. It contradicts with P_a that has the maximum number of *STR* from P_l. Therefore, P_l must exist in the current *STLC-DAG*.

Then, since P_l from STR_k^v to STR_i^j exists in *STLC-DAG*, it implies that there must exist a path P_b from the start vertex to STR_k^v in the current *STLC-DAG*. And, P_b contains all *STR* of *TR* between the start vertex and STR_k^v (P_c has passed through these *TRs*). This is because the algorithm has processed previous pair $\langle STR_t^r, STR_k^v\rangle$. And, there must exist a path P_t similar to P_a between STR_t^r and STR_k^v owing to the path found by *existPath*. And so on, these previous pairs form the P_b. Therefore, the P_b and P_l can form a spliced path. □

Lemma 5 If and only if a path found by algorithm 3 contains subtrajectories from two different trajectories, the two trajectories can be spliced.

Proof If there exists a path, which is found by Algorithm 3, between any two subtrajectories from two trajectories, respectively, according to Lemma 4, the trajectories that the path passed through can be spliced with the two subtrajectories. So, the two subtrajectories can be spliced. According to the definition 6, if two subtrajectories are spliceable subtrajectories, there exists a spliced path that can pass through all subtrajectories of the two trajectories. □

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

Proof Suppose there is an edge between STR_i^j and STR_m^n, which the two *STR* belong to TR_i and TR_j, respectively, and TR_i cannot be spliced with TR_m. According to Lemma 5, at least one pair of *STR* from the two *TR*, respectively, cannot be connected by a path that is found by $existPath$. But, as Algorithm 2 (Line 10) must have deleted all edges between TR_i and TR_j if it finds that a pair between them cannot be connected by a path. Therefore, there is not an edge between them. It contradicts the assumption that there is an edge between 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 As initialized phase of Algorithm 2, $SP = DT$. Suppose one SP_i has a subscript m, and its corresponding TR_m cannot be spliced with TR_i. According to Lemma 5, there is no a path between one pair $\langle STR_i^j, STR_m^n \rangle$. And, $SP_i = SP_i - m$ (Line 12 in Algorithm 2), has been executed. It contradicts with SP_i because SP_i contains m. □

Lemma 6. In *SP*-segraph, a clique is a group of spliceable trajectories, a maximal clique is a complete trajectory.

Proof A group of spliceable trajectories can be directly or indirectly spliced with each other. Therefore, there exists an edge between any two of them. So, the group of spliceable trajectories is a clique in the graph. If the clique is the maximal clique, the group of spliceable trajectories on the maximal clique cannot be contained by other groups. So, the maximal clique in the graph is a complete trajectory *CTR*. □

References

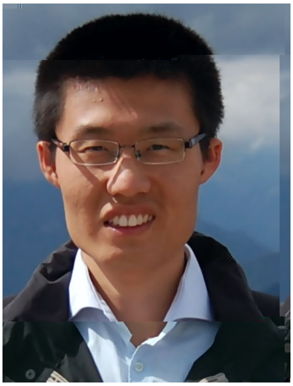
1. Bakalov P, Hadjieleftheriou M, Tsotras VJ (2005) Time relaxed spatiotemporal trajectory joins. In: Proceedings of the 13th annual ACM international workshop on geographic information systems, ACM, New York, NY, USA, pp 182–191
2. Dai J, Yang B, Guo C, Ding Z (2015) Personalized route recommendation using big trajectory data. In: 2015 IEEE 31st international conference on data engineering, pp 543–554
3. Dai J, Yang B, Guo C, Jensen CS, Hu J (2016) Path cost distribution estimation using trajectory data. Proc VLDB Endow 10(3):85–96
4. Ding Z, Yang B, Chi Y, Guo L (2016) Enabling smart transportation systems: a parallel spatio-temporal database approach. IEEE Trans Comput 65(5):1377–1391
5. Emrich T, Kriegel HP, Mamoulis N, Renz M, Züfle A (2012) Querying uncertain spatio-temporal data. In: 2012 IEEE 28th international conference on data engineering, pp 354–365
6. Eppstein D, Löffler M, Strash D (2010) Listing all maximal cliques in sparse graphs in near-optimal time. In: Algorithms and computation, no. 6506 in lecture notes in computer science, Springer Berlin Heidelberg, pp 403–414
7. Goh CH, Lu H, Ooi BC, Tan KL (1996) Indexing temporal data using existing B+-trees. Data Knowl Eng 18(2):147–165
8. Gudmundsson J, van Kreveld M (2006) Computing longest duration flocks in trajectory data. In: Proceedings of the 14th annual ACM international symposium on advances in geographic information systems, ACM, New York, NY, USA, pp 35–42
9. Gudmundsson J, van Kreveld M, Speckmann B (2004) Efficient detection of motion patterns in spatio-temporal data sets. In: Proceedings of the 12th annual ACM international workshop on geographic information systems, ACM, New York, NY, USA, pp 250–257
10. Guo C, Jensen CS, Yang B (2014) Towards total traffic awareness. SIGMOD Rec 43(3):18–23
11. Guo C, Yang B, Andersen O, Jensen CS, Torp K (2015) Ecomark 2.0: empowering eco-routing with vehicular environmental models and actual vehicle fuel consumption data. GeoInformatica 19(3):567–599
12. Guo C, Yang B, Hu J, Jensen CS (2018) Learning to route with sparse trajectory sets. In: IEEE 34th international conference on data engineering, pp 1073–1084
13. Güting RH, Valdés F, Damiani ML (2015) Symbolic trajectories. ACM Trans Spatial Algorithms Syst 1(2):7:1–7:51

19. Je ng H, Yi ML, Zho X, Jen en CS, Shen HT (2008) Di co e of con o in ajec o da aba e . 1:1068 1080
20. Kie T, Yang B, G o C, Jen en CS (2018a) Di ing i hing ajec o ie f om diffe en d i e ing incom le el labeled ajec o ie . In: P oceeding of he 27 h ACM in e na ional confe ence on info ma ion and kno ledge managemen , 863 872
21. Kie T, Yang B, Jen en CS (2018b) O lie de ec ion fo m l idimen ional ime e ie ing dee ne al ne o k . In: IEEE 19 h in e na ional confe ence on mobile da a managemen , 125 134
22. Kie T, Yang B, G o C, Jen en CS (2019) O lie de ec ion fo ime e ie i h ec en a oencode en emble . In: 28 h in e na ional join confe ence on a ificial in elligence
23. Ko e B, V gen J (2012) Combina o ial o imi a ion, algo i hm and combina o ic , ol 21. S inge , Be lin
24. Lee JG, Han J, Whang KY (2007) T ajec o cl e ing: a a i ion-and-g o f ame o k. In: P oceeding of he 2007 ACM SIGMOD in e na ional confe ence on managemen of da a, ACM, Ne Yo k, NY, USA, 593 604
25. Lee JG, Han J, Li X (2015) A nif ing f ame o k of mining ajec o a e n of a io em o al igh ne . IEEE T an 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 g o di co e in ajec o da aba e . IEEE T an Kno l Da a Eng 25(12):2752 2766
27. Li Z, Ding B, Han J, Ka R (2010) S a m: mining ela ed em o al mo ing objec cl e . P oc VLDB Endo 3:723 734
28. Liao L, Pa e on DJ, Fo D, Ka H (2007) Lea ning and infe ing an o a ion o ine . A if In ell 171(5 6):311 331
29. Sak MA, G ing RH (2011) S a io em o al a e n e ie . GeoInfo ma ica 15(3):497 540
30. S acca ie a S, Pa en C, Damiani ML, de Macedo JA, Po o F, Vangeno C (2008) A conce al ie on ajec o ie . Da a Kno l Eng 65(1):126 146
31. S H, Zheng K, H ang J, Wang H, Zho X (2014) Calib a ing ajec o da a fo a io- em o al imila i anal i . VLDB J 24(1):93 116
32. S n J, Tao Y, Pa adia D, Kollio G (2006) S a io- em o al join elec i i . Inf S 31(8):793 813
33. Tao Y, Pa adia D (2001) MV3 -T ee: A a io- em o al acce me hod fo ime am and in e al e ie . In: P oceeding of he 27 h in e na ional confe ence on e la ge da a ba e , Mo gan Ka fmann P bli he Inc., San F anci co, CA, USA, 431 440
34. Tomi a E, Tanaka A, Takaha hi H (2006) The o -ca e ime com le i fo gene a ing all ma imal cli e and com a ional e e imen . Theo Com Sci 363(1):28 42
35. Vald e F, G ing RH (2014) Inde - o ed a e n ma ching on mbolic ajec o ie . In: P oceeding of he 22Nd ACM SIGSPATIAL in e na ional confe ence on ad ance in geog a hic info ma ion em , ACM, Ne Yo k, NY, USA, 53 62
36. Viei a MR, Bakalo P, T o a VJ (2009) On-line di co e of flock a e n in a io- em o al da a. In: P oceeding of he 17 h ACM SIGSPATIAL in e na ional confe ence on ad ance in geog a hic info ma ion em , ACM, Ne Yo k, NY, USA, 286 295
37. Wang L, Zheng Y, Xie X, Ma WY (2008) A fle ible a io- em o al inde ing cheme fo la ge- cale GPS ack e ie al. In: 9 h in e na ional confe ence on mobile da a managemen , IEEE, 1 8
38. W H, X e M, Cao J, Ka a P, Ng WS, Koo KK (2016) F ajec o linking. In: IEEE 32nd in e na ional confe ence on da a enginee ing, IEEE, 859 870
39. Xie K, Deng K, Zho X (2009) F om ajec o ie o ac i i ie : a a io- em o al join a oach. In: P oceeding of he 2009 in e na ional o k ho on loca ion ba ed ocial ne o k , ACM, Ne Yo k, NY, USA, 25 32
40. X J, G ing RH, Zheng Y (2015) The TM-RT ee: an inde on gene ic mo ing objec fo ange e ie . GeoInfo ma ica 19(3):487 524
41. Yang B, Ma Q, Qian W, Zho A (2009) TRUSTER: ajec o da a oce ing on cl e . In: DASFAA, 768 771
42. Yang B, G o C, Jen en CS, Ka l M, Shang S (2014) S ocha ic k line o e lanning nde ime- a ing nce ain . In: IEEE 30 h in e na ional confe ence on da a enginee ing, 136 147
43. Yang B, G o C, Ma Y, Jen en CS (2015) To a d e onali ed, con e -a a e o 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 ic a adigm fo 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 e of ga he ing a e n o e ajec o ie . IEEE T an Kno l Da a Eng 26(8):1974 1988
46. Zheng Y (2015) T ajec o da a mining: an o e ie . ACM T an 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

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 users.



Bin Yang is a Professor in the Department of Computer Science at Aalborg University, Denmark. He was at Aarhus University, Denmark and at Max Planck Institute for Informatics, Germany. He received the Ph.D. degree in computer science from Fudan University. His research interests include machine learning and data management. He was a PC co-chair of IEEE MDM 2018. He has served on program committees and as an invited reviewer for several international conferences and journals, including ICDE, IJCAI, TKDE, the VLDB Journal, and ACM Computing Surveys.

Zhiguang Wang received a B.S. degree in physics from Inner Mongolia Normal University in 1986, an M.S. degree in computer engineering from Jilin University in 1994, and a Ph.D. degree in computer science from China University of Petroleum-Beijing. He is currently a Professor in the Department of Computer Science at China University of Petroleum, Beijing, he served as Director of Research Group of Large Scale Data Processing and Visualization. He is also a faculty member in Beijing Key Lab of Petroleum Data Mining, and a council member in Beijing Education Federation. His current research interests include data management, distributed systems, and spatial-temporal data mining.