Yin LH, Liu H. Searching activity trajectories with semantics. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 34(4): 775–794 July 2019. DOI 10.1007/s11390-019-1942-8

Searching Activity Trajectories with Semantics

Li-Hua Yin[1], *Member*, *CCF*, and Huiwen Liu[2,*]

[1] *Cyberspace Institute of Advanced Technology, Guangzhou University, Guangzhou 510006, China*
[2] *School of Information System, Singapore Management University, Singapore 188065, Singapore*

E-mail: yinlh@gzhu.edu.cn; hwliu.2018@phdis.smu.edu.sg

Received January 22, 2019; revised May 27, 2019.

Abstract With the widespread use of smart phones and mobile Internet, social network users have generated massive geo-tagged tweets, photos and videos to form lots of informative trajectories which reveal not only their spatio-temporal dynamics, but also their activities in the physical world. Existing spatial trajectory query studies mainly focus on analyzing the spatio-temporal properties of the users' trajectories, while leaving the understanding of their activities largely untouched. In this paper, we incorporate the semantics of the activity information embedded in trajectories into query modelling and processing, with the aim of providing end users more informative and meaningful results. To this end, we propose a novel trajectory query that not only considers the spatio-temporal closeness but also, more importantly, leverages a proven technique in text mining field, probabilistic topic modelling, to capture the semantic relatedness of the activities between the data and query. To support efficient query processing, we design a hierarchical grid-based index by integrating the probabilistic topic distribution on the substructures of trajectories and their spatio-temporal extent at the corresponding level of the index hierarchy. This specialized structure enables a top-down search algorithm to traverse the index while pruning unqualified trajectories in spatial and topical dimensions simultaneously. The experimental results on real-world datasets demonstrate the good efficiency and scalability performance of the proposed indices and trajectory search methods.

Keywords spatio-temporal database, activity trajectory, semantic understanding, trajectory indexing, trajectory query processing

1 Introduction

Trajectory data management has received increasing attention from both industries and academics in recent years. Prior work[1−8] has demonstrated the value of GPS trajectories for developing effective location-based recommendation systems (LBRS). With the widespread use of smart phones, social network users have generated massive geo-tagged records including geo-tagged tweets, photos and videos, which reveal not only their spatio-temporal dynamics but also their activities in the physical world. To model this enriched information, the concept of activity trajectory[9] has been proposed recently, which associates a set of activity terms (i.e., keywords) with each time-stamped location in the trajectories. Based on activity trajectories, a new trajectory query processing approach[9] has been formulated, which returns the most relevant activity trajectory to the user according to the user's query spatial location and textual intention. The new trajectory query processing requires the returned (sub-)trajectories to fully contain all the query activity terms and locate geographically close to the query locations. Although the intuitiveness of the query results has been shown in [9], the problem is obvious, that is, the required exact keyword match often leads to no or too few qualified results to be found.

To address the problem in [9], an immediate alternative is to apply the approximate keywords search with some text similarity measures (e.g., edit distance). This method can find trajectories with activity terms of similar spellings with respect to the query, which increases the robustness of the system for handling misspellings or spelling convention difference (e.g., center

Regular Paper
Special Section on Spatio-Temporal Big Data Analytics
This work was supported by the National Natural Science Foundation of China under Grant No. 61872100.
*Corresponding Author

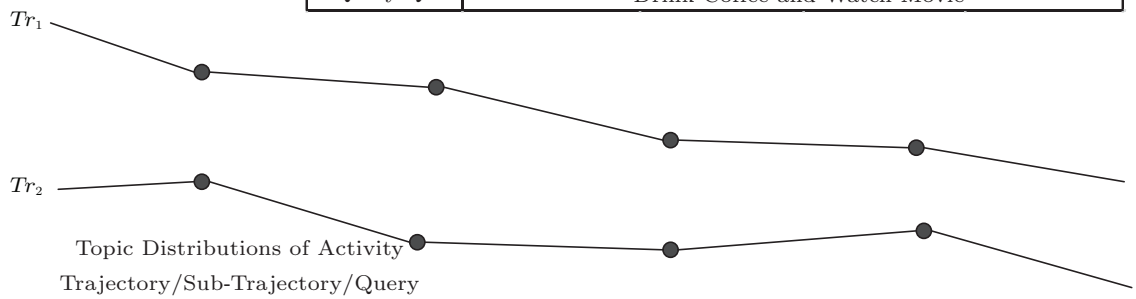
vs centre). However, it still cannot retrieve the activities with highly related semantics but low similarity in spellings, such as "Starbucks" and "coffee shop". Work [10] leverages the probabilistic topic model (e.g., LDA[11]), one of the proven successful techniques in text mining area, to replace the text similarity measure, and then improves the semantic matching accuracy effectively. However, whole trajectories rather than more meaningful sub-trajectories are used into query processing in [10], which severely deteriorates the efficiency of query responses and the performance of user experience. This limitation caused by the shallow segment of large span trajectory motivates us to investigate other approaches that aim to capture the semantic relatedness sub-trajectories.

Consider the example in Fig.1 where a tourist plans for a series of activities close to the location of Q and would like some recommendations based on the activity trajectories of other people in the surrounding area, such as Tr_1 and Tr_2, as references. Keyword-based trajectory search methods like [9] will only return Tr_2 as the result because it contains all the query keywords, though it is farther from Q than Tr_1. Meanwhile, semantic-aware trajectory search methods like [10] will return a more practical trajectory Tr_1 as the result, since "drink coffee" and "watch movie" (in Q) are quite related to "starbucks" (in $p_{1,1}$) and "cinema" (in $p_{1,2}$) in terms of their semantics, though their spellings share nothing in common. However, after examining those two trajectories more carefully, we find sub-trajectory $Tr_1^{1,2}$ is an even more relevant candidate to recommend since the rest points in Tr_1 (i.e., $p_{1,3}$ and $p_{1,4}$) are irrelevant to query Q in semantics. In order to provide smarter recommendation, the key problem is how to interpret and model the user's activity semantics based on the keywords, and then take the activity similarity at semantic level into consideration when searching the (sub-)trajectory database.

In this paper, we inherit the semantic model of trajectory activities in [10] to transform the activity terms into their semantic representations, which can then be used to quantify the semantic correlation between activities. For instance in Fig.1, by applying LDA model[11], we can obtain four latent topics and their distribution in Tr_1' is (0.55, 0.45, 0, 0), which indicates the relatedness between the overall activities of Tr_1' and each topic. Analogously, we can get the topic distribution for all the other trajectories (or even sub-trajectories) and the query. As all the activities are now represented by high-dimensional vectors (dependent on the number of topics), it is easy to measure the distance between topic distributions, which has turned out to be an effective measure for the thematic similarity between two documents in text mining field. By incorporating the distance on topics into the trajectory query processing, we can expect the results to have high spatio-

Keywords on Activity Trajectories Tr_1/Tr_2 and Query Q

	1st Point	2nd Point	3rd Point	4th Point
Tr_1	Starbucks	Cinema	Gym	Massage
Tr_2	Drink Coffee	Watch Movie	Body-Building	SPA
Query Q	Drink Coffee and Watch Movie			



Topic Distributions of Activity Trajectory/Sub-Trajectory/Query

	Café	Movie	Fitness	SPA
Tr_1	0.275	0.225	0.225	0.275
Tr_2	0.250	0.250	0.275	0.275
$Tr_{1'} = (p_{1,1}, p_{1,2})$	0.550	0.450	0.000	0.000
Q	0.500	0.500	0.000	0.000

Fig.1. Example of intention-oriented activity trajectory search.

temporal proximity and semantic correlation with respect to the given query.

Although the query semantic is easy to understand, answering this query efficiently is not straightforward. The reasons can be summarized into three folds. First, as a user is only interested in the part of a trajectory that is close and related to her/his query, it makes more sense to search sub-trajectories instead of whole trajectories. Consequently, the search space increases remarkably since a single trajectory contains a quadratic number of sub-trajectories. In addition, each sub-trajectory has its own topic distribution, which cannot be derived based on that of its parent trajectory. Second, the probabilistic topic distribution is calculated with an iterative algorithm, which is too expensive for online process, while on the other hand, the huge number of sub-trajectories prohibits the complete off-line calculation as well due to the high storage and the maintenance cost. Last but not least, the high dimensionality of vector representation of topics severely deteriorates the pruning effectiveness based on the information of subspaces (also known as the "curse of dimensionality"), which is critical for the efficiency of most multi-dimensional search algorithms.

Towards the topic model based activity trajectory/sub-trajectory search, we define a new type of trajectory similarity measure that considers both spatial and topic distances, and then propose a novel indexing structure called TP-tree. It keeps the advantage of hierarchical structure while avoiding the large dead space when indexing all the sub-trajectories in high dimensional spaces. To avoid storing quadratic number of sub-trajectory items in the index, long trajectories are partitioned into segments with good topic coherence guarantee, and only part of the sub-trajectories inside a segment appear as indexed items. Remaining sub-trajectories are represented by a limited number of merged clusters in index, so that the indexing structure can be controlled in a reasonable size. Particularly, we group similar nodes based on the estimation of their impacts to the dead spaces of all ancestor nodes, so that the tightness of the nodes can be fully guaranteed. Furthermore, a TP-tree based trajectory search algorithm is developed to efficiently process queries on scalable trajectories. The major contributions of this paper can be summarized as follows.

• We introduce and formalize a new type of topic model based similarity measure of activity trajectory for user intention interpretation, as well as intention-oriented recommendation.

• We develop a novel fine-grained index in a hierarchical structure to avoid not only the large "dead space" in the spatial and topic spaces (in high dimensional), but also the efficiency problems caused by the flaws of excessive (sub-)trajectory items in the index structure.

• We design an efficient intention-oriented trajectory search algorithm, which can effectively prune the search space to find the activity trajectory that can best satisfy the activity and spatial requirements of users.

• We conduct extensive experimental analysis based on real activity trajectory datasets, which includes the performance and efficiency comparisons.

The rest of the paper is organized as follows. Section 2 introduces the preliminaries and formally defines the activity trajectory search problem. Afterwards, we introduce some baseline methods in Section 3, and an improved trajectory partition based algorithm in Section 4. Section 5 reports the experimental observations. This paper is concluded in Section 6 after discussions on related work in Section 7.

2 Preliminaries and Problem Definition

This section provides the necessary definitions and gives the formal problem statement. Table 1 summarizes the notations used throughout the paper.

Table 1. Summary of Notations

Notation	Description
Tr	Activity trajectory
p	A point in an activity trajectory
$p.l$	Location of point p
$p.W$	Keywords attached on p to describe activity
$\kappa(Tr)$	Keywords of trajectory Tr
$Tr[i]$	The i-th point in trajectory Tr
$Tr_i^{j,k}$	Sub-trajectory of Tr_i from the j-th to the k-th point
$D_S(p, p')$	Spatial distance between points p, p'
$\boldsymbol{\beta}_w[z]$	Topic proportion of keyword w on topic z
$\boldsymbol{TD}_W$	Topic distribution of keywords W
$\boldsymbol{TD}_W[z]$	Topic proportion from W to topic z
$D_T(W, W')$	Topic distance between W and W'
$D(Q, Tr)$	Distance from query Q to trajectory Tr

2.1 Activity Trajectory

Definition 1 (Activity Trajectory). *An activity trajectory Tr is modeled as a sequence of geo-spatial points associated with textual descriptions about user activities, i.e., $Tr = (p_1, p_2, ..., p_n)$.*

Each point is in the form of $(l, tmsp, W)$, where l and $tmsp$ are the geographical location (in two-dimensional spaces) and the time-stamp respectively,

and W is the textual information formed by a set of keywords (in vocabulary V) that describes user activity at location l. We use $\kappa(Tr)$ to represent the keywords of trajectory, which is the union of the affiliated keywords on each point of Tr, and use $Tr[i]$ $(1 \leqslant i \leqslant |Tr|)$ to denote the i-th point on Tr.

An activity trajectory is essentially the historical records of the user activities and behaviors throughout a trip. In the rest of the paper, we will simply use trajectory to represent activity trajectory when no ambiguity can be caused.

Definition 2 (Sub-Trajectory). *$Tr_i^{j,k}$ is a sub-trajectory of trajectory Tr_i from its j-th point to the k-th point, denoted by $Tr_i^{j,k} \subseteq Tr_i$, if it holds that $Tr_i[n] \in Tr_i^{j,k}$ when $\forall n \in [j,k]$, and $Tr_i[n] \notin Tr_i^{j,k}$ when $\forall n \notin [j,k]$. We also use the term trajectory to represent sub-trajectory when no ambiguity can be caused.*

Definition 3 (Entry Point). *For a (sub-)trajectory $Tr_i^{j,k}$, we call the first point in $Tr_i^{j,k}$ (i.e., $Tr_i[j]$) as the entry point of $Tr_i^{j,k}$. If a user plans to follow $Tr_i^{j,k}$, he or she goes to its entry point firstly, and then continuously goes to the next point through the shortest path until the last point is reached.*

Definition 4 (Spatial Distance Measure). *Given a location p and a trajectory/sub-trajectory Tr, we use the sigmoid function to normalize their distance to the range $[0,1]$ as the following spatial distance measure:*

$$D_S(p, Tr) = \frac{2}{1 + e^{-dist(p,Tr[1]) - Length(Tr)}} - 1,$$

where $dist(p, Tr[1])$ is the distance from location p to the entry point of Tr (i.e., $Tr[1]$), and trajectory length $Length(Tr) = \sum_{1 \leqslant i \leqslant |Tr|} dist(Tr[i], Tr[i+1])$ is the total distance from the entry point to the last point in Tr. In other words, $dist(q.l, Tr[1]) + Length(Tr)$ is the total travel cost by following Tr. For example in Fig.1, given the spatial distance from $q.l$ to $p_{1,1}$ (e.g., 0.2 mile) and the trajectory length of Tr'_1 (e.g., 0.5 mile), then $D_S(q.l, Tr) = \frac{2}{1+e^{-0.7}} - 1$ is the spatial distance from the query location $q.l$ to the sub-trajectory Tr'_1.

2.2 Topic Distribution Based Distance Measure

Probabilistic topic model is a powerful tool for thematic interpretation of a large archive of documents over latent topics. By regarding the textual records in trajectories as documents and the thematic meanings as topics, we can apply a particular probabilistic topic model, e.g., Latent Dirichlet Allocation (LDA) model[11], to derive textual descriptions as distributions of topics. Here a topic can be understood as the thematic meaning of user activities as the following definition.

Definition 5 (Topic). *A topic z represents a type of activity that a user can take at some site of interest, such as "massage", "eating hotpot" and "movie". We use the finite $\mathbb{Z}$ to denote a pre-processed topic set, which is the union of all the topics related to user activity descriptions.*

How to generalize the topic vocabulary is an interesting research problem but it is outside the scope of this paper, and we assume that a given number of latent topics have already been extracted by some topic modeling algorithms. Based on statistical concurrence, each keyword in vocabulary has their probabilistic topic distributions over the latent topics, indicating their correlation between keywords and topics in thematic level.

Definition 6 (Topic Distribution of Keyword). *Given the $\mathbb{Z}$ topics, we use a matrix $\boldsymbol{\beta} = \mathbb{Z} \times V$ to describe topic distribution of a single keyword, where V is the given vocabulary of words. Each $\boldsymbol{\beta}_w$ is a distribution of keyword $w \in V$ over all topics, and $\boldsymbol{\beta}_w[z]$ is the topic proportion on topic z. For each keyword w, the sum of its topic proportion satisfies $\sum_{z \in \mathbb{Z}} \boldsymbol{\beta}_w[z] = 1$. Such a matrix $\boldsymbol{\beta}$ can be initialized by applying probabilistic topic models like LDA.*

Definition 7 (Topic Distribution of Keywords). *The topic distribution $\boldsymbol{TD}_W$ of multiple keywords W is the average of topic distributions of keywords in W, where the topic proportion $\boldsymbol{TD}_W[z]$ from W to topic z is calculated as*

$$\boldsymbol{TD}_W[z] = \frac{\sum_{w \in W} \boldsymbol{\beta}_w[z]}{|W|},$$

where $|W|$ is the number of keywords in W. The sum of topic proportions of any keywords in W is 1, i.e., $\sum_{z \in \rho(W)} \boldsymbol{TD}_W[z] = 1$. Given two sets of keywords W_1 and W_2, their aggregated topic proportion on $z \in \mathbb{Z}$ is computed as the average such that $TD_{W_1 \cup W_2}[z] = \frac{TD_{W_1}[z] + TD_{W_2}[z]}{2}$. The topic distribution $\boldsymbol{TD}_W$ is a vector in $|\mathbb{Z}|$ dimensions, and it can thus be seen as a point in a high-dimensional topic space, where each dimension is the thematic proportion on a topic. Topic distance between two keywords can thus be measured by the proportions in the topic space.

Definition 8 (Topic Distance Measure). *Given two keywords W and W', we use $D_T(\boldsymbol{TD}_W, \boldsymbol{TD}_{W'})$ to denote their topic distance which indicates their thematic similarity, and use the well-known Cosine distance to measure their topic distance in high-dimensional topic*

space, such that:

$$D_T(\boldsymbol{TD}_W, \boldsymbol{TD}_{W'}) = \sqrt{\frac{\sum_{z\in\mathbb{Z}} (\boldsymbol{TD}_W[z] - \boldsymbol{TD}_{W'}[z])^2}{|\mathbb{Z}|}}.$$

The greater topic distance, the less thematic relevance between the two given keywords. Obviously, such a measure limits the value of $D_T(p, p')$ to be in the range [0, 1]. It can be also expressed by $D_T(W, W')$ for simplicity.

Example 1. An example is shown in Table 2 to illustrate the topic distributions of textual descriptions in activity trajectories Tr_1 and Tr_2 (in Fig.1). The example consists of four topics, and the table includes the topic proportion of a keyword (e.g., $\boldsymbol{\beta}_{\text{“}starbucks\text{”}}[cafe] = 0.9$) and keywords (e.g., $\kappa(Tr_{1'}) =$ to be $TD_{\text{“}starbucks,cinema\text{”}}[cafe] = 0.55$), which indicate the thematic relevance between keyword(s) and topics. Based on topic distance measure, the topic distance between the query (i.e., $Q.W =$ “coffee and watch movie”) and the keywords of sub-trajectory $\kappa(Tr_{1'})$ (i.e., “starbucks, cinema”) is computed as $D_T(TD_{W_1}, TD_{W_2}) = 0.223$.

Table 2. Topic Distributions of Keyword(s) in Trajectories

Keyword(s)	Cafe	Movie	Fitness	SPA
Starbucks	0.90	0.10	0.0	0.0
Cinema	0.20	0.80	0.0	0.0
Gym	0.00	0.00	0.8	0.2
Massage	0.00	0.00	0.1	0.9
Coffee	0.90	0.10	0.0	0.0
Movie	0.10	0.90	0.0	0.0
Body-building	0.00	0.00	0.9	0.1
SPA	0.00	0.00	0.2	0.8
Starbucks and cinema	0.55	0.45	0.0	0.0
Coffee and watch movie	0.50	0.50	0.0	0.0

2.3 Problem Definition

Definition 9 (Trajectory Query). *A trajectory query $Q = (l, W)$ specifies the location of user l and a set of keywords W (e.g., “drink a cup of coffee, watch movie”) that describe the activity intention of users. We recommend an activity trajectory that is the closest to Q based on the following distance function.*

Definition 10 (Trajectory Distance Measure). *Given a trajectory query Q and an activity trajectory/sub-trajectory Tr, the trajectory distance from Q to Tr can be measured as the sum of distances between Q and Tr in both spatial and topic domains such that*

$$D(Q, Tr) = \lambda \times D_T(Q.W, \kappa(Tr)) + (1-\lambda) \times D_S(Q.l, Tr),$$

where $\kappa(Tr)$ is all keywords of Tr, and $\lambda \in [0, 1]$ is a user specified parameter used to adjust the relative importance of the topic similarity $D_T(Q.W, \kappa(Tr))$ and the spatial proximity $D_S(Q, Tr)$. Note that the topic distribution of Tr is subject to $\kappa(Tr)$, because part of trajectory keywords cannot reflect the real thematic meaning of the whole trajectory.

Definition 11 (Minimum Distance Match). *Given a query Q and a trajectory set τ, assume $ST = \{Tr_i^{j,k} \mid Tr_i \in \tau,\ 1 \leqslant i \leqslant j \leqslant |Tr_i|\}$ is the set of trajectories/sub-trajectories derived from trajectories in τ. A trajectory/sub-trajectory $Tr \in ST$ is said to be the minimum distance match to Q, denoted as $Tr.MDM(Q)$, if for any other trajectory/sub-trajectory $Tr' \in ST$ we have $D(Q, Tr) \leqslant D(Q, Tr')$. We call $D(Q, Tr)$ as the minimum match distance between Q and τ, denoted as $D_{\text{mm}}(Q, \tau)$.*

Problem Statement. Given an activity trajectory set τ, and a finite topic set $\mathbb{Z}$, a query Q, the user intention-oriented trajectory similarity query (UITSQ) in this paper is to return the (sub-)trajectory Tr from τ that is the minimum trajectory match (i.e., having minimum match distance) with respect to Q. For example in example 1, given a query $Q_1 =$ (l, “coffee and watch movie”), we aim to find the sub-trajectory $Tr_{1'}$ that is the minimum trajectory match to Q_1.

3 Baseline Algorithms

As far as we know, no existing methods can be directly applied for UITSQ. In this section, we propose two baseline algorithms to find practical solutions for this problem.

3.1 R-Tree Based Algorithm

In this algorithm, all trajectory points are treated as a point set, and the R-tree is used to index them to prune trajectories/sub-trajectories in pure spatial dimension. In the index structure, trajectory points are covered by the minimum bounded rectangles (MBR) based on spatial closeness, and all MBRs are organized in a hierarchical fashion. In processing a query Q, we incrementally traverse the index structure to find the nearest trajectory point in terms of spatial best match distance, where the spatial best match distance

$D_{\text{sbm}}(Q,p)$ between query Q and a trajectory point p is computed as

$$D_{\text{sbm}}(Q,p) = \lambda \times (\frac{2}{1+e^{-dist(Q.l,p)}} - 1).$$

For each selected trajectory point p, we derive the set of trajectories/sub-trajectories $TrS(p) = \{Tr \mid Tr[1] = p, Tr \in ST\}$ that regard p as a trajectory entry point. Accordingly, the spatial best match distance is a lower bound of the trajectory distance from Q to trajectories/sub-trajectories in $TrS(p)$. As shown in the following lemma, we can prune out far-away trajectories based on their entry point in spatial dimension to speed up query processing.

Lemma 1. *$\lambda \times (\frac{2}{1+e^{-dist(Q.l,p)}} - 1)$ is a lower bound of trajectory distance $\lambda \times (\frac{2}{1+e^{-dist(Q.l,p)}} - 1) \leqslant D(Q,Tr)$ for each $Tr \in TrS(p)$.*

Proof. Given that p is the first point in trajectory Tr, we have $\lambda \times (\frac{2}{1+e^{-dist(Q.l,p)}} - 1) \leqslant D_S(Q,Tr) \leqslant D(Q,Tr)$. □

Lemma 1 gives us the opportunities to prune the search space. Similar to the BCT query in [5], we keep finding the next nearest trajectory point p, and compute the trajectory distance $D(Q,Tr)$ from query Q for each trajectory/sub-trajectories $Tr_i^{j,k}$ having $Tr_i[j] = p$ in an incremental fashion. A global upper bound of trajectory distance D_{UB} is defined as the smallest trajectory distance of the partly scanned trajectories/sub-trajectories such that

$$D_{UB} = \min_{Tr_i^{j,k} \in T_s}\{D(Q,Tr_i^{j,k})\},$$

where T_s is the set of sub-trajectories we have scanned. Obviously, D_{UB} is dynamically updated in search processing, and it is the threshold in query processing: if a trajectory point p is as far as $D_{\text{sbm}}(Q,p) > D_{UB}$ already, all trajectories having $Tr[1] = p$ are hopeless to be better than $D_{\min}$ according to Lemma 1. As trajectory points are retrieved in ascending order of spatial best match distance, the query processing can be stopped if a trajectory point p can be found such that $D_{\text{sbm}}(Q,p) > D_{UB}$, because all remaining trajectories can be pruned by the long distance from $Q.l$ to their entry point (i.e., the first trajectory point).

For each non-pruned trajectory point $Tr_i[j]$, i.e., $D_{\text{sbm}}(Q,p) \leqslant D_{UB}$ is true, we conduct the following operations in loop: 1) selecting an un-processed trajectory $Tr_i^{j,k} \in TrS(p)$ in the ascending order of k; 2) computing the trajectory distance $D(Q,Tr_i^{j,k})$, and terminating the loop if $\lambda \times (\frac{2}{1+e^{-dist(Q.l,p)-Length(Tr_i^{j,k})}} - 1) \geqslant D_{UB}$ because remaining sub-trajectories have no possibility to improve the current threshold according to Lemma 2.

Lemma 2. *If the spatial distance of a sub-trajectory $Tr_i^{j,k}$ holds that $\lambda \times D_S(Q.l,Tr_i^{j,k}) > D_{UB}$, then all the sub-trajectories $Tr_{j,k'}^{i}$ $(k < k')$ can be pruned from the candidate set.*

Proof. Given that $D_S(Q.l,Tr_{j,k'}^{i}) > D_S(Q.l,Tr_i^{j,k})$ and $D_S(Q.l,Tr_i^{j,k}) \geqslant D_{UB}$, it must be true that $D(Q,Tr_{j,k'}^{i}) > \lambda \times D_S(Q.l,Tr_{j,k'}^{i}) \geqslant D_{UB}$. □

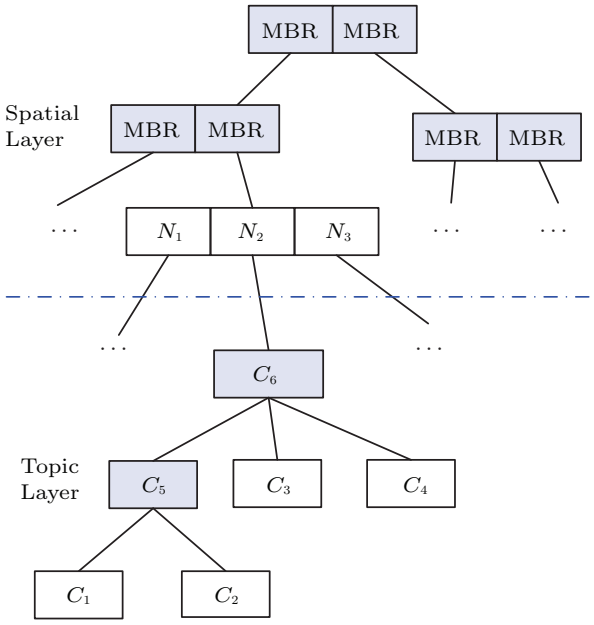
Example 2. In Fig.1, let us consider two trajectories $Tr_a = (p_{1,1}, p_{1,2})$ and $Tr_b = (p_{1,1}, p_{1,2}, p_{1,3})$. For trajectory point $p_{1,1}$, if the condition $\lambda \times (\frac{2}{1+e^{-dist(Q.l,p_{1,1})}} - 1) \geqslant D_{UB}$ is true, then both Tr_a and Tr_b can be pruned because of the long distance to their entry point $p_{1,1}$ according to Lemma 1. If condition $\lambda \times (\frac{2}{1+e^{-dist(Q.l,p)-Length(Tr_a)}} - 1) \geqslant D_{UB}$ can be satisfied in trajectory search, trajectory Tr_b can be pruned because its spatial distance $D_S(Q.l,Tr_b)$ must be higher than upper bound distance D_{UB} based on Lemma 2.

In overall, the pruning of this algorithm is only restricted in spatial domain, leading the performance of query processing to be impeded especially when the value of parameter λ is small. We further design a hybrid index structure to integrate information in both spatial and topic domains.

3.2 Hybrid Indices Based Algorithm

The basic idea of this algorithm is to model each trajectory or sub-trajectory as a point in high-dimensional space regarding to their spatial and topic features. A hybrid trajectory index structure called Topic Retrieval R-tree (TR^2-tree) is designed to achieve effective pruning in both spatial and topic domains.

Index Structure. The overall structure of the TR^2-tree is shown as Fig.2. Basically, TR^2-tree has two layers: the spatial layer in above, and the topic layer in bottom. In the spatial layer, we use R-tree to index all trajectory points like the first algorithm. Each leaf node in R-tree links to a topic layer indexing component, in which all trajectories/sub-trajectories relevant to this R-tree leaf node are indexed in the topic space. A trajectory or sub-trajectory is said to be relevant to an MBR node if its entry point is inside this node. For example in Fig.2, the sub-trajectory ($p_{1,1}$, $p_{1,2}$) is relevant to the node MBR_1 because its entry point $p_{1,1}$ is covered by MBR_1. For each MBR node, all relevant trajectories can be seen as a point (based on $TD_{\kappa(Tr)}$) in topic space with $|\mathbb{Z}|$ dimensions. The problem is how



Spatial Layer Nodes

Spatial Node	Items	Child Node
N_1	$\{p_{1,2}, p_{2,2}\}$	
N_2	$\{p_{1,1}, p_{2,1}\}$	C_6
N_3	$\{p_{1,3}, p_{2,4}\}$	

Topic Layer Nodes

Cluster	Items	Reference Point	MD_i
C_1	$\{(p_{1,1}), (p_{1,2}), (p_{1,1}, p_{1,2})\}$	(0.55, 0.45, 0, 0)	0.494 9
C_2	$\{(p_{1,2}, p_{1,3}), (p_{1,1}, p_{1,2}, p_{1,3})\}$	(0.234, 0.35, 0.33, 0.086)	0.161 0
C_3	$\{(p_{1,3}), (p_{1,4}), (p_{1,3}, p_{1,4})\}$	(0, 0, 0.45, 0.55)	0.495 0
C_4	$\{(p_{1,1}, p_{1,2}, p_{1,3}, p_{1,4}), (p_{1,2}, p_{1,3}, p_{1,4})\}$	(0.275, 0.225, 0.225, 0.275)	0.246 0
C_5	$\{C_1, C_2\}$	(0.392, 0.4, 0.165, 0.043)	0.632 7
C_6	$\{C_3, C_4, C_5\}$	(0.334, 0.312, 0.358, 0.363)	0.753 2

Fig.2. Index structure of TR^2-tree.

to index a large number of high-dimensional points in the topic layer effectively.

In the topic layer, we utilize the iDistance[12] indexing method to index the topic distribution (i.e., high-dimensional vectors) of scalable activity trajectories. The purpose of utilizing the iDistance index is to speed up trajectory search by topic domain pruning. For all trajectories/sub-trajectories in an R-tree leaf node, a partition method K-medoids is applied to group all trajectories into k clusters first based on their distribution in topic space (i.e., $TD_{\kappa(Tr)}$). Each cluster C_i ($i \in [1, k]$) covers trajectories that refer to similar topics (e.g., C_1, C_2 and C_3 in Fig.2) by a reference point m_i (in the topic space) and a maximum distance MD_i to m_i such that

$$MD_i = \max(\sqrt{\sum_{z\in\mathbb{Z}}(TD_{\kappa(Tr)} - m_i[z])^2} \mid Tr \in C_i).$$

Therefore, in TR^2-tree, each leaf node in the topic layer is a cluster in the form of $C_i = (m_i, MD_i)$, where m_i and MD_i are the reference point and the maximum distance of the cluster respectively. The form of each point in a cluster is $p = (i, j, k, TD_{\kappa(Tr)})$, which refers to a sub-trajectory $Tr_i^{j,k}$, and $TD_{\kappa(Tr)}$ denotes the topic distribution with $|\mathbb{Z}|$ dimensions. Then the cluster nodes are further organized into a hierarchical structure as shown in Fig.2: the reference point m_i of each non-leaf node C_i is the center of all reference points of its child nodes; the maximum distance is computed as $MD_i = \max_{C_j \in CN}(dist(m_i, m_j) + MD_j)$, where CN is the set of child nodes of C_i, and $dist(m_i, m_j)$ is the distance between reference points m_i and m_j in the high-dimensional topic space. The space represented by m_i and MD_i obviously covers the topic distribution of all sub-trajectories which regard C_i as their ancestor node.

Trajectory Search. As the TR^2-tree structure tends to be large, the trajectory search is conducted in the spatial and topic domain alternatively. As shown in Algorithm 1, we start from the spatial layer and access R-tree leaf nodes in ascending order of $SD_{\min}$, which is the minimum spatial distance from query location $Q.l$ to the MBR of this node (line 3). If $\lambda \times (\frac{2}{1+e^{-SD_{\min}}} - 1) > D_{UB}$ can be satisfied, all remaining trajectories are pruned by the spatial proximity (line 5). Otherwise we go to the topic layer, and access child clusters of the R-tree node in ascending order of their best match distance to the query, where the best match distance $D_{\mathrm{bm}}(Q, C_i)$ between the query Q and a cluster C_i is computed as

$$D_{\mathrm{bm}}(Q, C_i) = \lambda \times (\frac{2}{1 + e^{-SD_{\min}}} - 1) + (1 - \lambda) \times |D_T(TD_{Q.W}, m_i) - MD_i|,$$

where $SD_{\min}$ is the minimum spatial distance between the query location $Q.l$ and the MBR (in spatial layer) of

the R-tree node that covers C_i. As shown in Lemma 3, the best match distance of clusters is a lower bound of the trajectory distance $D_S(Q, Tr)$ for any $Tr \in C_i$, so that all trajectories in a cluster can be pruned if the lower bound of distance is greater than that of the current best trajectory D_{UB} already.

Algorithm 1. TR^2-Tree Based Trajectory Search

```
Input: τ: activity trajectory set; Q: query;
λ: importance factor
Output: Tr: trajectory
1  D_UB = +∞; Tr = NULL
2  foreach leaf R-tree node N in ascending order of SD_min do
3      if λ × (2/(1+e^{-SD_min}) − 1) > D_UB then
4          break;
5      else
6          foreach leaf-cluster C_i ∈ N in ascending order of
           D_bm(Q, C_i) do
7              if D_bm(Q, C_i) < D_UB then
8                  for p = (i, j, k, TD_{κ(Tr)}) ∈ C_i do
9                      if D(Q, Tr_i^{j,k}) < D_UB then
10                         Tr = Tr_i^{j,k};
11                         D_UB = D(Q, Tr)
12             else
13                 break;
14 return Tr;
```

Lemma 3. *The best match distance $D_{\mathrm{bm}}(Q, C_i)$ is a lower bound of the trajectory distance between an arbitrary trajectory in cluster C_i and query Q.*

Proof. Assume Tr is an arbitrary trajectory in C_i, we have: 1) in the spatial domain, as $SD_{\min} \leqslant dist(Q.l, Tr[1])$, it holds that $\lambda \times (\frac{2}{1+e^{-SD_{\min}}} - 1) \leqslant D_S(Q.l, Tr)$; 2) in the topic domain, as MD_i is the maximum topic distance to reference point m_i, it must hold that $|D_T(Q.W, m_i) - MD_i| \leqslant D_T(Q.W, m_i) - D_T(\kappa(Tr), m_i) \leqslant D_T(Q.W, \kappa(Tr))$ in high-dimensional topic space. By putting spatial and topic domain together, we can thus conclude that $D_{\mathrm{bm}}(Q, C_i)$ is the lower bound of $D(Q, Tr)$ of any trajectory $Tr \in C_i$ according to 1) and 2). □

If the best match distance of a cluster C_i is greater than the threshold $D_{\mathrm{bm}}(Q, C_i) > D_{UB}$, all trajectories in this cluster can be pruned directly (line 14). For non-pruned clusters, we validate the sub-trajectory information and update D_{UB} if a better sub-trajectory is found (line 11). It is obvious that the search space can be greatly reduced by the pruning in both spatial and topic spaces in the collaborative search.

4 Trajectory Partition Based Algorithm

To improve the TR^2-tree based algorithm by reducing excessive indexed items, we propose a novel trajectory partition based indexing structure called TP-tree. The basic idea is to partition an activity trajectory Tr into segments if $|Tr|$ is high, and index only part of the sub-trajectories that can be wholly covered by one trajectory segment. For remaining activity sub-trajectories, we derive the bounded topic spaces that can cover their topic distributions, so that effective pruning can be achieved finally. In Subsection 4.1, we discuss how to partition long trajectories in rational. Based on partitioned segments, the fine-grained index structure is introduced in Subsection 4.2, and then Subsection 4.3 describes the efficient search over scalable trajectories/sub-trajectories.

4.1 Trajectory Partition

As each sub-trajectory becomes an index item in TR^2-tree, the general purpose of trajectory partition is to reduce the number of items in leaf nodes of trajectory data index. If a trajectory Tr has a high value of $|Tr|$, we partition Tr_i into a sequence of segments S_{i1}, S_{i2}, ..., S_{in}, where $S_{i1} \cup S_{i2} \cup ... \cup S_{in} = Tr_i$ and $S_i \cap S_j = \emptyset$ $(1 \leqslant i \neq j \leqslant n)$. Each segment is a concessive part of Tr that consists of trajectory points in Tr defined as follows.

Definition 12 (Trajectory Segment). *A trajectory segment is a concessive part of a trajectory.*

Trajectory segments are basic units of the TP-tree. Given a trajectory Tr_i, we use $S_i^{j,k}$ to denote a segment from $Tr_i[j]$ to $Tr_i[k]$ after partition, and use $S_i^{j,k}[x]$ to denote the x-th $(1 \leqslant x \leqslant k - j + 1)$ point inside segment $Tr_i[j, k]$, which can be mapped to the trajectory as $S_i^{j,k}[x] = Tr_i[j + x - 1]$.

In the TP-tree, each segment appears as a leaf node, and only the sub-trajectories inside one segment become an indexed item in the corresponding leaf node. Therefore, the performance of TP-tree is adaptive to trajectory partition. In trajectory partition, we consider two main issues as constraints in partition.

• To limit the index size, each segment is allowed to have trajectory points no greater than b (i.e., $\forall S_i^{j,k}$: $k - j < b$), and it thus generates up to $\ulcorner \frac{|Tr|}{b} \urcorner \times \frac{(1+b)\times b}{2}$ items in indexing structure, where $\ulcorner \frac{|Tr|}{b} \urcorner$ is the maximum number of segments of a trajectory Tr and $\frac{(1+b)\times b}{2}$ is the number of sub-jectories in a segment with the points number b.

• To avoid excessive cluster nodes in TP-tree, the number of trajectory points in each segment must be greater than a threshold a, because each segment becomes a leaf node (i.e., cluster node) in TP-tree.

We define the minimum bounding topic space (MBTS) as the minimum space (similar to MBR in two-dimensional spaces) to cover a trajectory segment or cluster in topic domain. Given a segment $S_i^{j,k}$, its corresponding size of MBTS, denoted as $size(S_i^{j,k})$, is computed as

$$size(S_i^{j,k}) = \prod_{z\in\mathbb{Z}} |1 + MaxTP(S_i^{j,k}, z) - MinTP(S_i^{j,k}, z)|,$$

where $MaxTP(S_i^{j,k}, z)$ and $MinTP(S_i^{j,k}, z)$ return the maximum and the minimum topic proportion to z of any sub-trajectories in segment $S_i^{j,k}$ respectively. Obviously, the value of $size(\mathbb{S})$ indicates the dead space, and it subjects to the trajectory points in the segment only. Given an arbitrary set of trajectory segments $\mathbb{S}$, the total MBTS of the set is computed as the sum of segment MBTSs such that

$$size(\mathbb{S}) = \sum\nolimits_{S_i^{j,k}\in\mathbb{S}} size(S_i^{j,k}).$$

In trajectory partitioning, we intend to minimize the total MBTS size of the set of concessive partitioned segments after partition, because less total dead space contributes to effective pruning in trajectory search.

Therefore, in this paper, we seek for the trajectory partitioning mechanism that can efficiently find the segment set $\mathbb{S}$ of a trajectory such that: 1) the value of $size(\mathbb{S})$ is to be minimized for effective pruning; 2) the number of trajectory point in each segment $S_i^{j,k} \in \mathbb{S}$ is ensured to be in the range of $[a, b]$. A segment $S_i^{j,k}$ is said to be valid only if $|S_i^{j,k}| \in [a, b]$.

To find the set of trajectory segments $\mathbb{S}$ mentioned above, we use a dynamic programming based approach to partition each trajectory into trajectory segments as required. To find the optimal partition (in the global point of view) of a trajectory Tr_i on the last segment as $f(n)$, it depends on the partition of previous segment according to $f(n-x)$. Therefore the basic idea is to find optimal partitioning point iteratively based on the cost function $f(n)$ defined as follows.

$$f(n) = \begin{cases} \min\{f(n-x) + \\ \qquad size(S_i^{n-x+1,n})\}, & \text{if } n \geqslant a, \\ 0, & \text{if } n = 0, \\ +\infty, & \text{if } n \in (0, a), \end{cases}$$

where $x \in [a, b]$ indicates a partitioning point on $Tr_i[n-x]$, and $size(S_i^{n-x+1,n})$ is the MBTS of the last segment $S_i^{n-x+1,n}$, thereby we have $f(n) = f(n-x) + size(S_i^{n-x+1,n})$ for arbitrary last partitioning point x. $f(n) = +\infty$ if $n \in (0, a)$ because the segment is invalid. In $f(n-x)$, we further compare the last partitioning points in an iterative way based on the function. In this way, all possible trajectory partitions are compared (note that some of them can be simply filtered in implementation) to find the optimal partitioning solution as required. Based on the trajectory segments after partition, we construct the well-sized indexing structure to support efficient activity trajectory search.

For each long trajectory, we further propose a heuristic-based method to partition them efficiently. The trajectory is continuously partitioned until all segments become valid regarding to $[a, b]$ constraint. In each time, we select a segment with more than b trajectory points first, and partition it into two segments that incur minimum increase of $size(S)$. For each invalid segment S_{ij}, we derive its valid partitioning point candidate set $VP(S_{ij})$, which allows each sub-trajectory to be able to become valid segments by additional partitions. Then we compare the points in $VP(S_{ij})$ to select out the trajectory point that leads to the minimum value of total size $size(S)$ after the partition. The above partition is carried out on invalid segments in loop, until all segments become valid segments (i.e., the size constraint can be satisfied).

Index Structure. The TP-tree is a two-layered index as shown in Fig.3. The upper layer is the spatial layer, in which each trajectory is covered by an MBR that covers all trajectory points. All MBRs are further organized in hierarchy like R-tree. Note that each leaf node corresponds to one activity trajectory in the spatial layer only. For each trajectory, a tree-based structure is utilized to manage its sub-trajectory topic distributions. The key challenge is to avoid enumerating topic distributions of all sub-trajectories.

We propose a trajectory segment based indexing mechanism to address the above challenge. Given a trajectory, each of the partitioned segment becomes a segment node in the topic layer, such as nodes S_1–S_4 in Fig.3. The form of each sub-trajectory $Tr_i^{j,k}$ in a segment is $(i, j, k, TD_{\kappa(Tr_i^{j,k})})$, and a grouping mechanism is applied to convert them into clusters. Each cluster becomes a cluster node similar to the TP-tree (with a space in topic domain described by m_i, MD_i), and it is a child node of the segment node it belongs to. Towards two adjacent segments (e.g., S_1/S_2 and S_3/S_4),

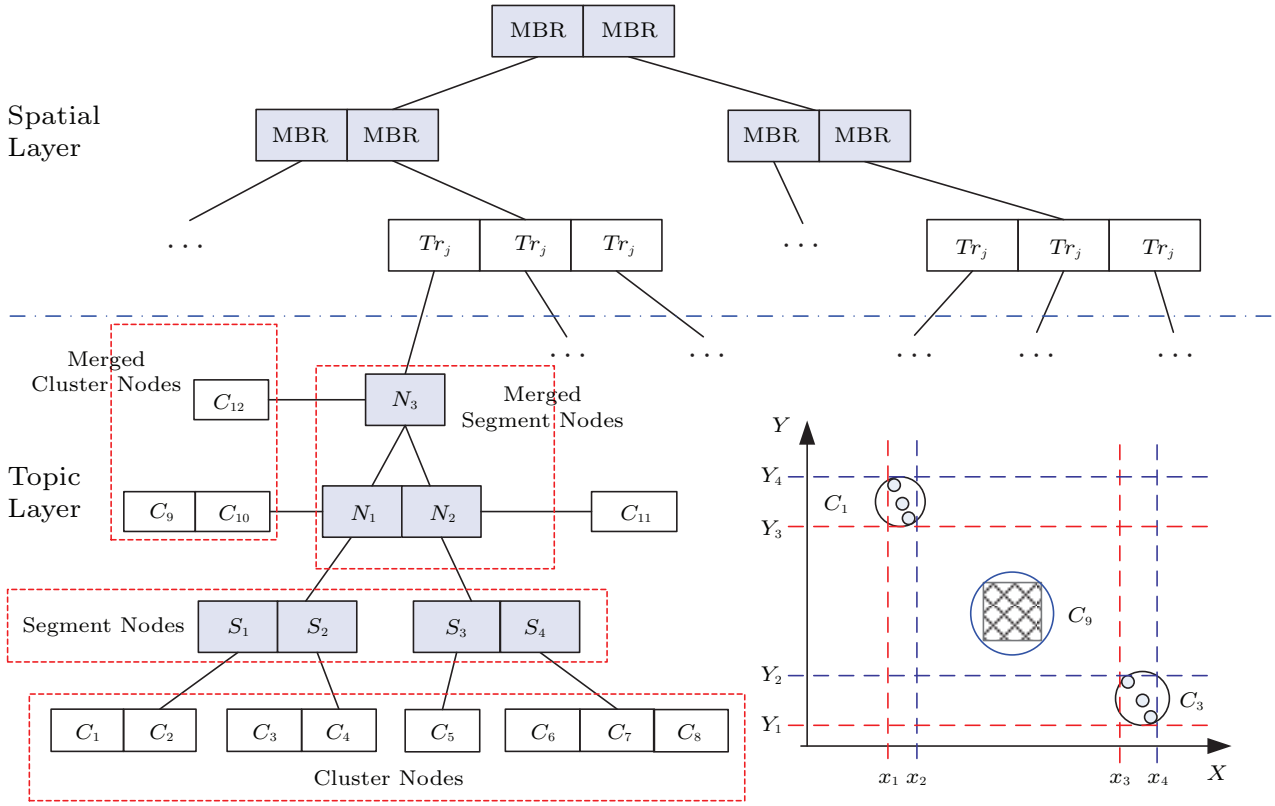


Fig.3. Index structure of TP-tree.

we create a merged segment node (e.g., N_1) to denote the segment consisted by them. The merge operation is made in hierarchy until a root is found. The form of both segment nodes and merged segment nodes is (ID, m_i, MD_i, C), where reference point m_i and maximum distance MD_i together denote a high-dimensional topic space that can cover all child nodes in C.

For each merged segment node like N_1, it is important to associate the node to some fine-grained spaces to summarize all sub-trajectories of this segment in the topic domain. But the union of child node space used in TP-tree cannot be applied, because cross segment sub-trajectories may not be covered in any child node. As such, we further discuss how to compute the fine-grained spaces to cover all sub-trajectories of merged segment nodes like N_1, N_2 and N_3.

Merging Segment Nodes. In merging adjacent segment nodes, additional sub-trajectories cross two segments will be generated, and the key issue is to represent their topic distributions in some nodes by sketch. We investigate how to assign each merged cluster with a set of find-grained topic spaces that can cover all merged sub-trajectories (i.e., intersecting to both segments). Particularly, it is carried out in such a way that only the information in the involved child nodes is used for efficiency guarantee. Before discussing how to merge the cluster nodes, we first define the notion of "mergeable" nodes as follows to determine if two cluster nodes can be merged.

Definition 13 (Mergeable Nodes). *Given two nodes a and b in the topic layer, they are said to be mergeable only if there exist two concessive sub-trajectories $Tr_i^{j,k} \in a$ and $Tr_i^{k+1,n} \in b$ respectively.*

If two segment nodes are mergeable, their merge must generate new sub-trajectory intersecting to both segments, and then a merged cluster (e.g., C_9) must appear in the index to represent those merged sub-trajectories by a small topic space that can cover them. Given two mergeable cluster nodes C_i and C_j $(i < j)$, we can derive the minimum bounding topic space $SP = \{(SP[z].min, SP[z].max) | z \in \mathbb{Z}\}$ that can cover all merged sub-trajectories such that

$$SP[z].min = \frac{W_{\max}(C_i) \times V_{\min}[C_j][z] + W_{\min}(C_j) \times V_{\min}[C_i][z]}{W_{\max}(C_i) + W_{\min}(C_j)},$$

$$SP[z].max = \frac{W_{\max}(C_j) \times V_{\max}[C_i][z] + W_{\min}(C_i) \times V_{\max}[C_j][z]}{W_{\min}(C_i) + W_{\max}(C_j)},$$

where $W_{\min}(C_i)$ and $W_{\max}(C_i)$ return the minimum and the maximum number of trajectory points of any trajectory in the node C_i respectively. Also we have $V_{\min}[C_i][z]$ and $V_{\max}[C_i][z]$ to return the minimum and the maximum value of possible topic proportion to z of the given C_i, such that

$$V_{\min}[C_i][z] = \min(\{TD_{\kappa(Tr)}[z]|Tr \in C_i\}),$$
$$V_{\max}[C_i][z] = \max(\{TD_{\kappa(Tr)}[z]|Tr \in C_i\}).$$

We maintain two matrix structures $\boldsymbol{V}_{\min}$ and $\boldsymbol{V}_{\max}$ to store the maximum and the minimum topic proportions of the nodes that have not been involved in nodes merge. For each merged cluster or segment node, two lists are used to store the entry point set and the end point set of all associated sub-trajectories, so that their mergeable nodes can be verified. The topic distribution of any merged sub-trajectory by C_i and C_j can be covered by the space SP according to Lemma 4.

Lemma 4. *In topic domain, space SP covers the topic distributions of all merged sub-trajectories by C_i and C_j.*

Proof. Assume that $Tr_a^{b,d}$ is an arbitrary sub-trajectory merged by $Tr_a^{b,c} \in C_i$ and $Tr_a^{c+1,d} \in C_j$, and its topic proportion on a given topic z can be computed as

$$TD_{\kappa(Tr_a^{b,d})}[z] = \frac{|Tr_a^{b,c}| \times TD_{\kappa(Tr_a^{b,c})}[z] + |Tr_a^{c+1,d}| \times TD_{\kappa(Tr_a^{c+1,d})}[z]}{|Tr_a^{b,c}| + |Tr_a^{c+1,d}|}.$$

Given that $W_{\min}(C_i) \leqslant |Tr_a^{b,c}| \leqslant W_{\max}(C_i)$ and $W_{\max}(C_j) \leqslant |Tr_a^{c+1,d}| \leqslant W_{\min}(C_j)$, it must hold that $SP[z].min \leqslant TD_{\kappa(Tr_a^{b,d})}[z] \leqslant SP[z].max$ for $\forall z \in \mathbb{Z}$, so that SP must cover $TD_{\kappa(Tr_a^{b,d})}$ in the topic space. □

Then we use a reference point m_k and a maximum distance MD_k to represent the covering space SP in the merged cluster node C_k. The reference point is set as the central of SP, with a proportion to each topic z to be $m_k[z] = \frac{SP[z].max - SP[z].min}{2}$. In SP, the points in $\{p|p[z] = SP[z].min \vee p[z] = SP[z].max)\}$ have equal distance to m_k, all larger than the distance from other points to m_k, and the proof is omitted due to the limit of space. Therefore, the maximum distance is set as $MD_k = \sqrt{\frac{\sum_{z \in \mathbb{Z}} (SP[z].max - m_k[z])^2}{|\mathbb{Z}|}}$ to cover the topic distributions of all merged sub-trajectories together with m_k.

Example 3. A simple case (involving two topics for clear description) in Fig.3 illustrates how the nodes are merged. Assume nodes C_1 and C_3 are mergeable, and a merged cluster C_9 is created to represent their merged sub-trajectories. Towards the mergeable nodes, we apply the above formula to derive the space (in black) that covers the topic distribution of any sub-trajectory merged through C_1 and C_3, and the space is further transformed by a reference point (the center) and a maximum distance (the radius) expression in C_9. Similarly, C_{10} can be derived based on the mergeable nodes C_2 and C_4. In constructing the index, our key purpose is to make the space of nodes to be tight, and we discuss how to support that by rational grouping of sub-trajectories in leaf nodes.

Grouping. We group sub-trajectories (to form cluster nodes) to reduce overall dead space based on the scope of whole index structure. Given a group of sub-trajectories to appear in a cluster node C_j, the dead space $DS(C_i)$ is measured as the MBTS to cover all sub-trajectories such that

$$DS(C_j) = \prod_{z \in \mathbb{Z}} |1 + SP_{C_j}[z].max - SP_{C_j}[z].min|,$$

where SP_{C_j} is the MBTS of the cluster C_j, and $SP_{C_j}[z].max$ is the maximum value of topic proportion from a sub-trajectory in C_j to the topic z. The basic criterion of the grouping is similar to the R-tree: given a trajectory Tr that is partitioned into a set of segments S, we group sub-trajectories in each segment to minimize the total dead space $DS(S_i) = \sum_{S_i \in S} \sum_{C_j \in S_i.succ} DS(C_j)$, where $S_i.succ$ returns the child nodes of S_i. But the sub-trajectories cross different segments are represented by the merged segment nodes, not in $DS(Tr)$ yet. Therefore additionally, we must notice its impact to the dead spaces of nodes in a higher level (via segment merge).

Firstly, the grouping of sub-trajectories impacts the total number of ancestor nodes (i.e., merged cluster nodes). Given a segment S_i, we need to particularly pay attention to the merge points denoted by $mp(S_i)$, i.e., the breaking trajectory points (i.e., the first or last) after trajectory partition, because $mp(S_i)$ incurs additional merged cluster nodes according to Lemma 5. Obviously, $mp(S_i)$ is the first trajectory point if S_i merges with the previous segment S_{i-1}, or the last point if S_i merges with S_{i+1}.

If there exists a sub-trajectory in a cluster C_j (e.g., C_1 in Fig.3) that contains $mp(S_i)$, then C_j must be mergeable with at least one node in the other segment to incur merged a cluster node (e.g., C_9) according to Lemma 5, and we call it a merge candidate. For example in Fig.3, C_1 finally incurs a merged cluster node C_9 because $mp(S_i)$ belongs to at least one sub-trajectory

in C_1. We use $MC(S_i)$ to denote the set of merge candidates in S_i (invoking merged cluster nodes), and it is expressed as follows.

$$MC(S_i) = \{C_j | C_j \in S_i.succ \wedge \exists Tr_a^{b,c} \in C_j : mp(S_i) \in Tr_a^{b,c}\}.$$

In the grouping of sub-trajectories, the less $|MC(S_i)|$ is preferred to reduce the number of merged cluster nodes in the TP-tree.

Lemma 5. *A cluster C_j in segment S_i incurs at least one merged cluster node in the TP-tree if there exists a sub-trajectory $Tr_a^{b,c} \in C_j$ that holds $mp(S_i) \in Tr_a^{b,c}$.*

Proof. When we merge segments S_i and S_m ($m = i+/-1$), there will be at least one cluster $C_k \in S_m$ that contains a sub-trajectory $Tr_{a'}^{b',c'}$ that holds the merge point $mp(S_m) \in Tr_{a'}^{b',c'}$. As $Tr_{a'}^{b',c'}$ and $Tr_a^{b,c} \in C_j$ can merge to a new sub-trajectory, C_j and C_k are mergeable and they together generate a merged cluster node in the TP-tree. □

Secondly, the grouping mechanism has impacts on the dead spaces of the parent nodes. The MBTS of parent nodes is computed by that of its child nodes according to previous section, and obviously, the space would be smaller if the clusters have consistent values of $W_{\min}(C_i)$ and $W_{\max}(C_i)$. Given a merge candidate node C_i, we intend to minimize $ND(C_i) = W_{\max}(C_i) - W_{\min}(C_i)$ to ensure the tightness of the merged cluster nodes incurred by grouping.

Therefore, it is crucial to integrate above impacts and carefully design the grouping mechanism for effective pruning in trajectory search. In this paper, we apply a heuristic grouping method to balance different factors. For each segment in the trajectory, we first derive all sub-trajectories without containing the merge point, and then apply the K-medoids clustering to group them into k clusters (each has a reference point and a radius) based on their topic distributions. For each remaining sub-trajectory $Tr_i^{a,b}$ that covers the merge point, we measure the penalty of grouping it with a cluster C_j by extending the R-tree measure as

$$PNT(Tr_i^{a,b}, C_j) = \triangle DS(C_j) \times \triangle MC(C_j) \times \sqrt{\triangle ND(C_j)},$$

where $\triangle DS(C_j)$, $\triangle MC(C_j)$ and $\triangle ND(C_j)$ are the differences between the initial values of $DS(C_j)$, $MC(C_j)$ and $ND(C_j)$ to those values after we group $Tr_i^{a,b}$ to C_j respectively. For example, assume that $DS(C_j) = 0.2$ initially and $DS'(C_j) = 0.25$ if a sub-trajectory is grouped to C_j, then $\triangle DS(C_j) = 0.25$. For each remaining sub-trajectory that covers the merge point of segment, we insert it to the cluster with minimum penalty, and then update the related information on the corresponding node. This procedure is continued until all sub-trajectories in the segment have been inserted.

4.2 Trajectory Search

As shown in Algorithm 2, we adopt a priority queue based trajectory search to avoid unnecessary searching efforts. Specifically, our method incrementally finds out the most promising trajectory candidates, and validates the relevant segments of each trajectory candidate. In the searching process, we maintain the minimum trajectory distance D_{UB} found so far, which is the upper bound of the minimum match distance $D_{\text{mm}}(Q, \tau) \leqslant D_{UB}$. Above procedure repeats until remaining candidates are hopeless to improve D_{UB} (i.e., all worse than the current best result).

Algorithm 2. TP-Tree Based Trajectory Search

```
   Input: τ: activity trajectory set; Q: query
   Output: Tr: trajectory
1  D_UB = +∞; Tr = NULL
2  while true do
3      TC ← retrieve μ new spatial layer leaf nodes;
4      foreach N_i ∈ TC in ascending order of D_sm(Q, N_i) do
5          NQ ← N_i.succ; //* the queue of nodes to check
6          SC = NULL;      //* segment candidate
7          if N_i.succ is a segment node then
8              if D_m(Q, N_i.succ) < D_UB then SC ← N_i.succ
9          while NQ is not empty do
10             N_j = dequeue(NQ)
11             foreach merged segment node N_k ∈ N_j.succ do
12                 if D_m(Q, N_k) < D_UB then NQ ← {N_k}
13             foreach segment node N_k ∈ N_j.succ do
14                 if D_m(Q, N_k) < D_UB then SC ← {N_k}
15             foreach cluster node C_k ∈ N_j.succ do
16                 if D_m(Q, C_k) < D_UB then
17                     Add relevant segment nodes to SC;
18             Update NQ and SC based on nodes in N_j.succ;
19         foreach Tr_i^{b,c} covered by SC do
20             if D(Q, Tr_i^{b,c}) < D_UB then
21                 Tr = Tr_i^{b,c}; D_UB = D(Q, Tr_i^{b,c});
22     if ∃N_i ∈ TC : D_sm(Q, N_i) > D_UB then
23         break;
24 return Tr;
```

Firstly, we incrementally traverse the spatial layer of the TP-tree to retrieve the best-μ trajectory candi-

dates, according to the spatial closeness between $Q.l$ and the MBRs of trajectories (line 3). Specifically, we traverse the nodes in the spatial layer, and maintain a dynamic priority queue PQ with entries in the form $(i,\ D_{\mathrm{sm}}(Q, N_i))$, where $D_{\mathrm{sm}}(Q, N_i)$ is the spatial minimum distance from the query Q to any sub-trajectory covered by the node N_i such that

$$D_{\mathrm{sm}}(Q, N_i) = \lambda \times (\frac{2}{1 + e^{-SD_{\min}(Q, getMBR(N_i))}} - 1),$$

where $SD_{\min}(Q, getMBR(N_i))$ returns the minimum spatial distance from $Q.l$ to the MBR of the node N_i. The nodes with minimum spatial distance will put on the top of the heap. Then similar to the best-first search, we repeatedly dequeue the top entry of PQ. If its child nodes are spatial layer nodes, we insert them to the proper location in PQ. Otherwise (assume it denotes a trajectory Tr_i), the trajectory becomes one of the μ trajectory candidates in set TC. The dequeue operation stops when a number of μ trajectory candidates are collected.

Secondly, we further validate sub-trajectories in set TC of μ trajectory candidates. A brute force method is to calculate $D(Q, Tr_i^{j,k})$ of each sub-trajectory in TC, thereby it validates $\sum_{Tr_i \in TC} \frac{(1+|Tr_i|)\times|Tr_i|}{2}$ sub-trajectories. To improve efficiency, we adopt a divide-and-conquer method to prune sub-trajectories in non-relevant segments. For each spatial layer leaf node $N_i \in TC$ (assume denoting trajectory Tr_i), we start from its root node in the topic layer (e.g., N_3 in Fig.3) to find the set SC of segment candidates relevant to query (lines 7–18). The node N_i represents the trajectory if it is a segment node, and thus we insert it to SC directly if $D_{\mathrm{m}}(Q, N_j) < D_{UB}$. Otherwise, we traverse and validate its child nodes by a queue NQ, which is used to store the nodes in topic layer (initially containing $N_i.succ$). For each node $N_j \in NQ$, the minimum distance between N_j and the query Q can be calculated as $D_{\mathrm{m}}(Q, N_j) = \lambda \times (\frac{2}{1+e^{-SD_{\min}}} - 1) + (1 - \lambda) \times |D_T(TD_{Q.W}, m_j) - MD_j|$.

The validation of segment candidates is made for each node in the queue NQ. For each top node N_j in NQ derived by dequeue (line 11), we update NQ and SC based on each node in $N_j.succ$: when N_j is a merged segment node, the child node is inserted to SC if $D_{\mathrm{m}}(Q, N_j) < UB$, because the sub-trajectories in this segment have potential to be better than the current best result. Otherwise N_j is a merged segment node. When N_j is merged by merged segment nodes, the child nodes are inserted to NQ if their minimum distances to query are less than D_{UB}, and are pruned otherwise. When N_j is merged by segment nodes, we insert them to SC directly if and only if their minimum distances to query are less than D_{UB}.

Then we check all of the sub-trajectories that fall in SC (lines 19–21). For the segment nodes in SC, we simply check all sub-trajectories from the pre-computed information of its child nodes (i.e., clusters). For the merged segment nodes (e.g., N_1 in Fig.3) in SC, we retrieve the sub-trajectories cross the covered segments (i.e., S_1 and S_2) from the trajectory data. If the distance is less than D_{UB}, we update the result candidate and D_{UB} accordingly. Above processing (lines 3–23) is conducted in loop until we reach a node $N_i \in TC$ that satisfies $D_{\mathrm{sm}}(Q, N_i) > D_{UB}$ because remaining trajectories have no chance to improve the current best trajectory (lines 22–24).

5 Experimental Study

We conduct extensive experiments on the real activity trajectory datasets to evaluate the performances of our proposed algorithms.

5.1 Experimental Settings

We use a real activity trajectory dataset by crawling the geo-tagged social media messages from the biggest Chinese social media website, Sina Weibo[①]. Each message contains a user ID, geo-location located in Beijing, time and a tweet describing the user's activities. We treat the tweet as a bag of keywords and LDA tool is used to convert the keywords into a topic distribution (over 100 latent topics). We treat the messages belonging to the identical user as his or her activity trajectory. The generated dataset from Sina Weibo includes 723 628 activity trajectories, and the number of trajectory points is 10 971 353 in total (i.e., around 15 points per trajectory in average).

Then we compare the proposed algorithms in different test settings based on the dataset. The default values of test settings are summarized in Table 3. In our experiments, we vary these values to investigate their effects on the performances of algorithms. For each set of experiments, we generate 100 queries (by random spatial location and topic distributions) and report the average CPU time and I/O cost. All algorithms are implemented in JAVA and tested on a HP Compaq 8180 Elite (i5 650) computer with 2-core CPUs at 3.2 GHz

①https://www.weibo.com, May 2019.

and 1.12 GHz, 8 GB RAM and running Windows XP operating system. In the proposed index structures, we set each non-leaf node to have up to 40 child nodes and each leaf node to correspond to a 4 KB disk page by default.

Table 3. Topic Distributions of Keyword(s) in Trajectories

Parameter	Default Value	Description
$\|\tau\|$	1×10^4	Number of trajectories
$\|Tr\|$	4×10^1	Number of points in each trajectory
λ	5×10^{-1}	Weight factor
$\|\mathbb{Z}\|$	1×10^2	Number of latent topics

5.2 Performance Evaluation

Effect of $|\tau|$. To evaluate the scalability performance, we first compare all algorithms in different numbers of activity trajectories. To achieve that, we sample the dataset to generate datasets with different numbers of trajectories varying from 50k to 400k. The CPU time and the I/O cost of three algorithms are reported in Fig.4. According to Fig.4(a), the TR^2-tree based algorithm has the minimum CPU time. It can be explained by the fact that all sub-trajectory information is stored in database and scanned in query processing. On the other hand, it makes TR^2-tree have the high I/O cost in Fig.4(b), as excessive data need to be loaded into memory. In comparison, the TP-tree based algorithm is the most efficient solution because of its good I/O performance according to Fig.4(b), and saves more CPU time than the TR^2-tree based approach as well. Its superior I/O performance is due to two reasons: 1) the size of the TP-tree is much smaller than that of the TR^2-tree; 2) it retrieves each activity trajectory from disk only once. The R-tree based algorithm has the worst performances on both CPU time and I/O cost because of the naive pruning strategy in spatial domain only and multiple disk retrieval for each trajectory.

Effect of $|Tr|$. We also evaluate the performance of the proposed algorithms on trajectories with different numbers of trajectory points. For the sampled 100k trajectories (they hold $|Tr| \geqslant 100$), we generate datasets with different numbers of trajectory points varying from 20 to 100, and report performances of the proposed methods in Fig.5. From the figures we can observe that the R-tree based algorithm has the most stable performance in terms of the I/O cost, because it is a point-based indexing method, while the other two algorithms try to enumerate sub-trajectories. However, this means the R-tree based algorithm has much greater CPU time than the others. In comparison, the TP-tree based algorithm has the best overall performance: it has less I/O cost than TR^2-tree and R-tree based algorithms in Fig.5(a) because only the information of sub-trajectories within a segment is recorded, and slightly greater CPU time than TR^2-tree as shown in Fig.5(b).

Effect of λ. Fig.6 shows the CPU time and I/O cost of the proposed algorithms in a different weight factor λ. We can observe that all algorithms are sensitive to the value of λ, and they basically have the similar trend, i.e., the CPU time and I/O cost of all algorithms reduce when λ goes up, notably a sharp decrease happens when λ is less than 0.8. This phenomenon can be explained by the spatial first nature of the proposed algorithms, which leads to better pruning effects on the pure spatial queries (the TP-tree organizes the MBRs

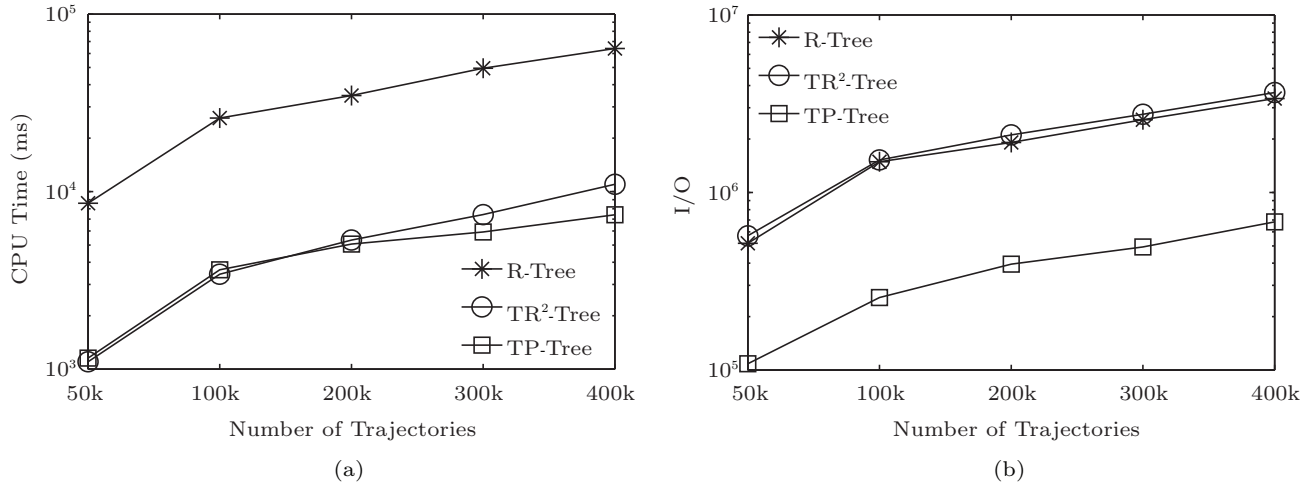


Fig.4. Effect of $|\tau|$. (a) CPU time comparison. (b) I/O cost comparison.

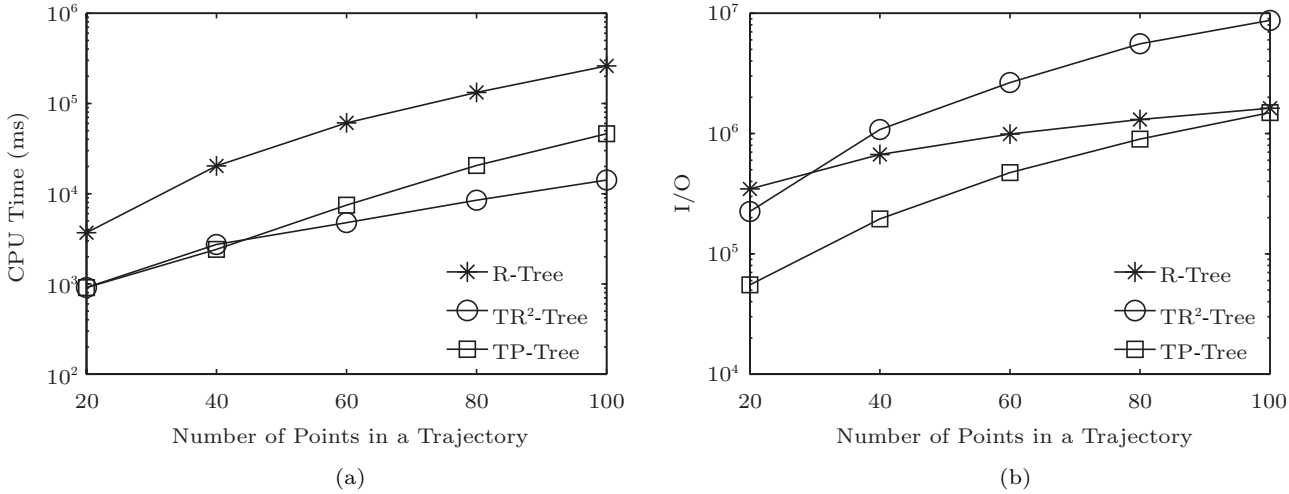


Fig.5. Effect of $|Tr|$. (a) CPU time comparison. (b) I/O cost comparison.

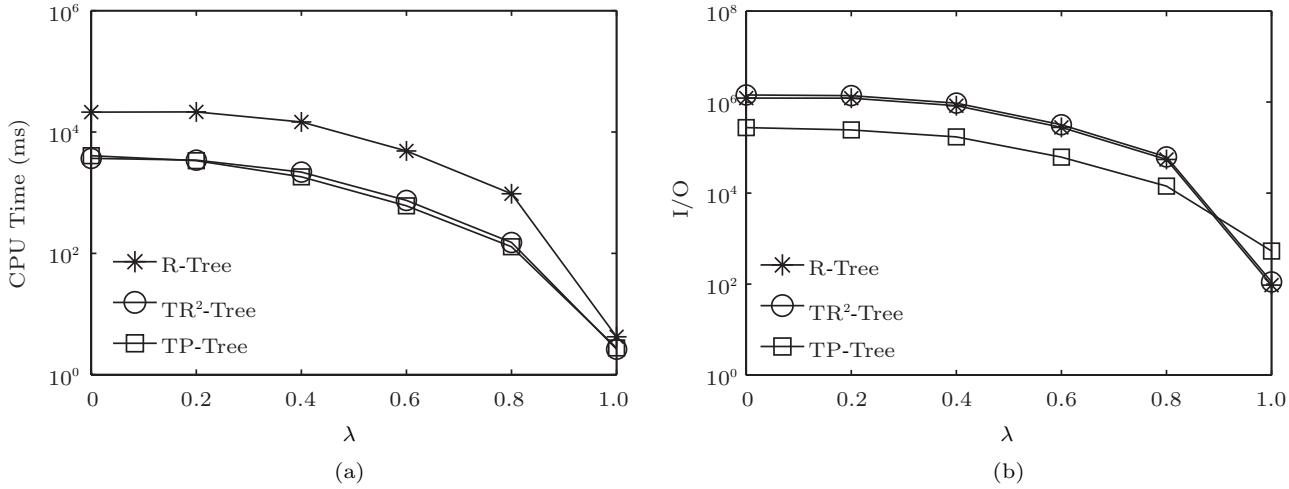


Fig.6. Effect of $|\lambda|$. (a) CPU time comparison. (b) I/O cost comparison.

in trajectory granularity, thereby it has more I/O cost when $\lambda = 1$). Overall, the TP-tree based algorithm is superior to the other two algorithms with less I/O cost and CPU time, and the fluctuation is more stable than the others as well. Therefore in comparison, TR^2-tree generally outperforms the R-tree in all λ settings, but all worst with the TP-tree based strategy as shown in Fig.6.

Effect of Partitioning and Grouping Mechanisms. Then we further evaluate the effect of long trajectory partitioning approach of the TP-tree based algorithm. The experiment is made on a varying number of activity trajectories from 10k to 50k, and all trajectories have a number of 100 trajectory points. Fig.7 reports the performances by adopting different parameters of partitioning. The experimental results show that the smaller size of segment incurs more CPU time and less I/O cost, which coincides with the mechanism of the TP-tree based approach such that having more segments tends to reduce the storage cost but incurs extra computations during the trajectory search process.

In addition, the effect of grouping mechanism of TP-tree based algorithm is also tested. Fig.8 shows the CPU time and I/O cost of TP-tree based approach with and without using the proposed heuristic grouping mechanism, respectively. It is apparent that the heuristic grouping based approach has less CPU time and I/O cost from Fig.8(a) and Fig.8(b) respectively. This phenomenon demonstrates that the grouping mechanism contributes to reducing the number of merged cluster nodes as well as the search space. Also, Fig.9 proves the effectiveness of the segment sketch method in the

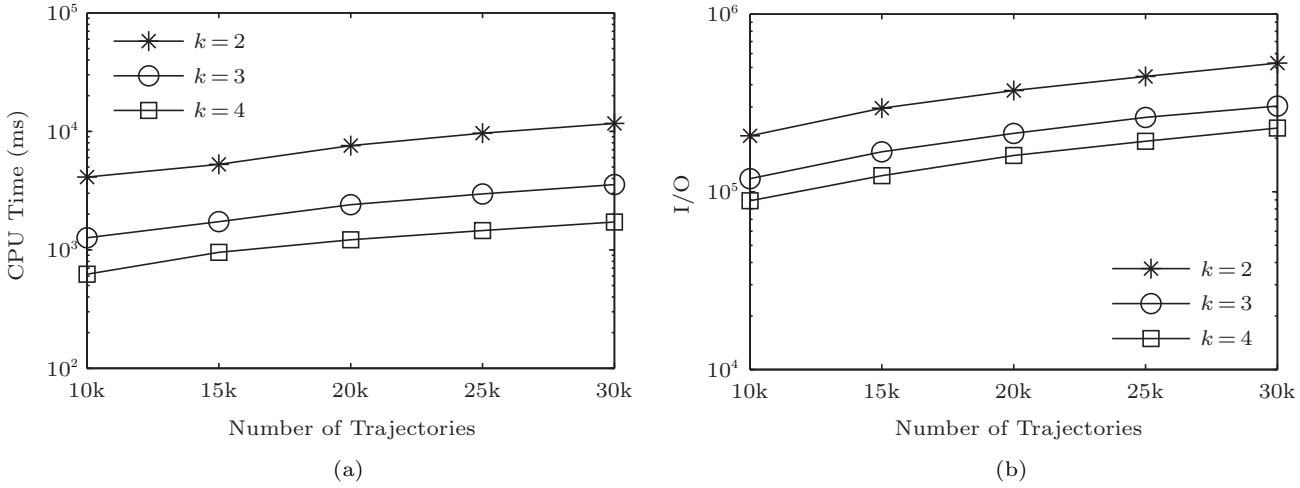


Fig.7. Effect of trajectory partitioning. (a) CPU time comparison. (b) I/O cost comparison.

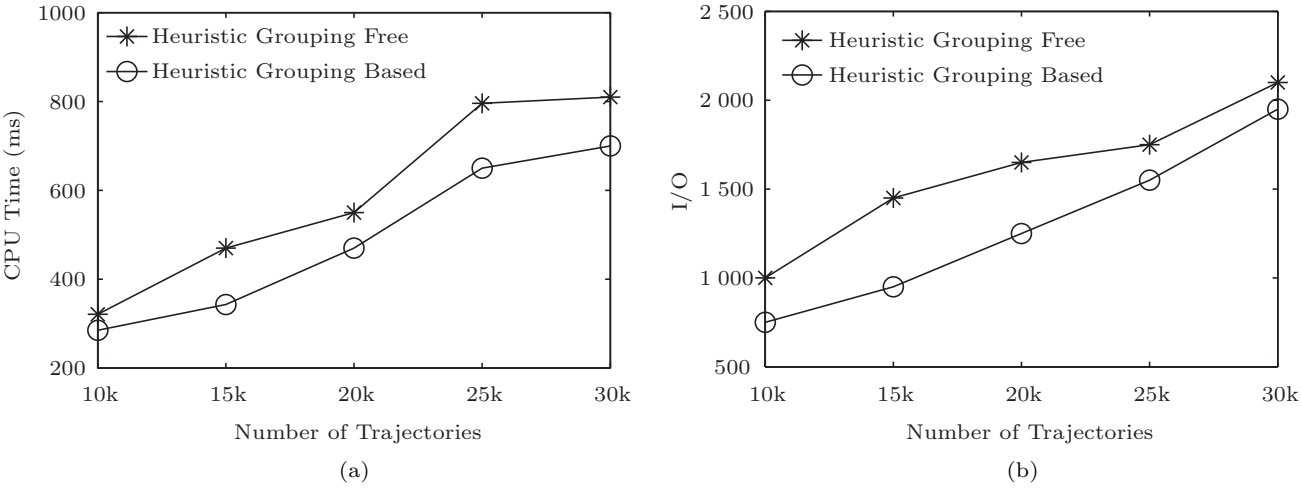


Fig.8. Effect of grouping. (a) CPU time comparison. (b) I/O cost comparison.

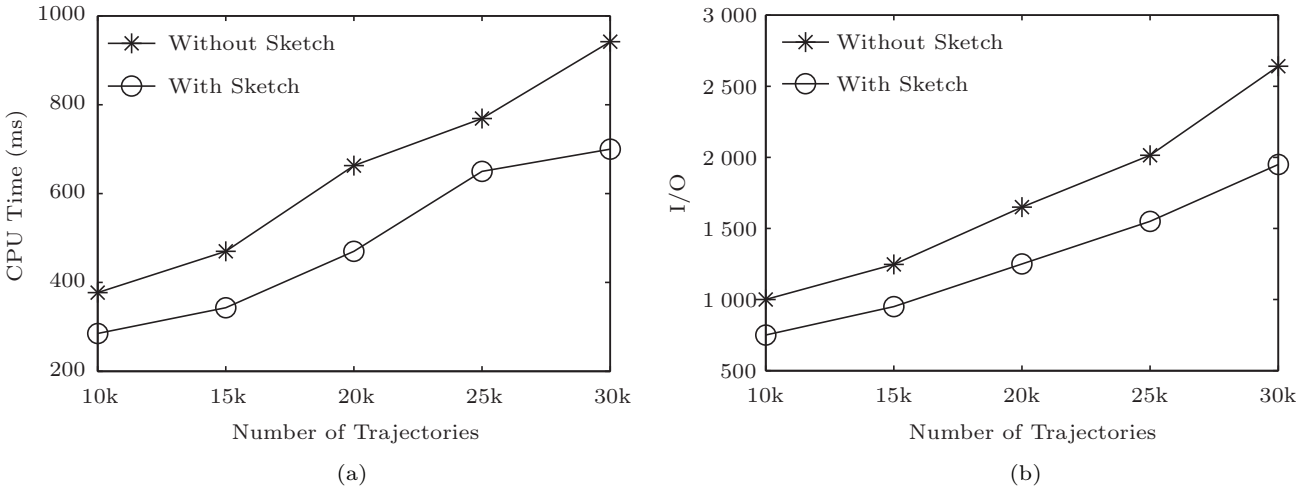


Fig.9. Effect of segment sketch. (a) CPU time comparison. (b) I/O cost comparison.

activity trajectory search processing.

Comparisons on Updating Cost. We also evaluate the performance of the proposed methods in inserting an activity trajectory in the varying trajectory number and trajectory length, the results of which are reported in Fig.10. In comparison, the TR^2-tree has the worst updating performance because it cannot support incremental updates (i.e., any insertion, update, or delete operation requires to reconstruct the index), and thus it is not reflected in Fig.10. From Fig.10(a), we can observe that TP-tree requires less time except for long trajectory insertion, which is caused by the higher total costs on the topic domain clustering of sub-trajectories in the TP-tree for a long trajectory. Fig.10(b) shows that the TP-tree has the best overall update performance among all three proposed algorithms.

To sum up, the TP-tree based index structure and searching algorithm has better querying processing performance in all of the settings (subject to dataset and query) because of the lower I/O cost and CPU time. In addition, it incurs the least update cost due to the way in which the trajectory data are organized in the index. The trajectory partitioning and grouping mechanisms contribute to guarantee the good performance of TP-tree.

6 Related Work

The related work to our study includes probabilistic topic model, spatial keyword search, and trajectory search.

Probabilistic Topic Model. The probabilistic topic models are statistical methods to analyze the words in documents and to discover the themes that run through them, which is used to interpret how those themes are connected to each other with no prior annotations or labeling of the documents being required[13]. Based on the topic models, it is possible to measure the relevance between a text and a theme, as well as that between different texts[14]. The most classical topic models include LDA[11], Dynamic Topic Model[15], Dynamic HDP[16], Sequential Topic Models[17], Poisson Decomposition Topic Model[18], etc. Moreover, some efforts have been made to extend the classic topic model of keywords to the topic model of phrase[19], and even the sentence[20] to consider the cohesive and sequential factors of the keywords in the sentence, which can effectively improve the accuracy of the semantic understanding. With the improvement of the semantic accuracy, some studies began to apply the topic models in location-based service recommendation systems[21−23]. Above techniques have been widely used in applications like document classification, user behavior understanding, functional region discovery. In this paper, we tend to integrate topic model and activity trajectory or user intension oriented trajectory search.

Spatial Keyword Search. Searching spatial objects associated with textual information has gained significant attentions due to the prevalence of spatial web objects on the Internet. Well known methods like IR-tree[24], S2I[25], RCA[26], IR^2-tree[27], SI-index[28], inverted Grid index[29], and NPD-index[30] were proposed to study the spatial keyword search problem in high dimension. More recently, lots of efforts have been made to handle spatial keyword search on road network[31−33], collective querying[34], confiden-

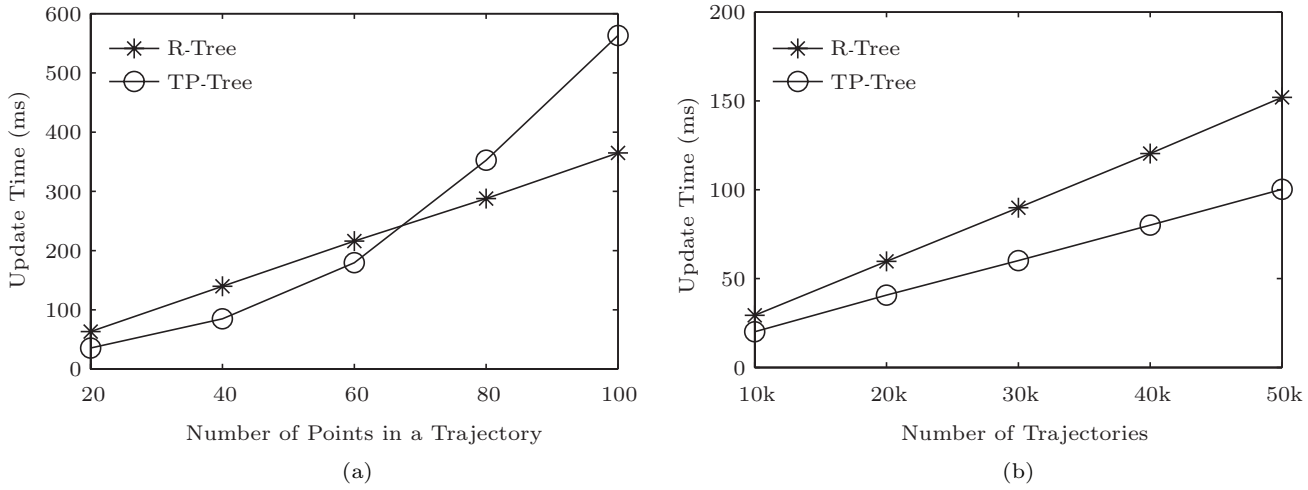


Fig.10. Updating cost. (a) Varying trajectory points. (b) Varying the number of trajectories.

tiality support[35], social network[36] and continuous querying[37,38]. Above methods are mature enough to handle the misspellings or spelling convention difference in spatial object search, and even under road network constraints. But on the other hand, they are not capable of retrieving spatial objects with highly related semantics but low similarity in spellings. To address this issue, studies[39,40] proposed the semantic-based spatial keyword querying, which integrates the semantic understanding to the objects querying processing. But both of the two studies only took the static spatial object into account, rather than dynamic activity trajectory. In this paper, we investigate the topic model based activity trajectory search to better satisfy user intentions.

Trajectory Search. The problem of (sub-)trajectory similarity search has been extensively studied in the last two decades[1−5,41−45]. Initially, trajectory search is mostly constrained in the spatial and temporal domains only. To achieve better utilization of trajectory data, research efforts have also been made to transform a raw GPS trajectory into a semantic trajectory[46,47], which consists of a sequence of stops at places of interest (PoIs) labelled with semantic tags (e.g., category description). Since the location tags imply the potential activities being carried out, semantic trajectory analytics can reveal the high-level semantic behaviors of users[48] and facilitate location-activity recommendations[49,50]. But the erroneous mappings could happen in those approaches even if they are spatially close enough. More recently, some academic efforts have been made to deal with the trajectory search in high-dimensional space. The work of [51] proposed effective indexing and efficient trajectory algorithm for queries that have specified their query location and user-preference attributes (e.g., "by bus"). In [9], a novel indexing structure is proposed to speed up the search of trajectory that can cover the required activities in query based on a given activity vocabulary. But this approach can only find the ones containing all the desired activities at each query location, which results in no or few results and thus does not fulfill the requirements of recommendation. The work of [10] proposed semantic-aware query processing which takes the semantic similarity into the activity matching. But this approach considers whole trajectory, including semantic irrelevant trajectory points. Furthermore, the approaches in [9, 10, 51] cannot be directly applied to our topic model based trajectory search problem because of the much greater scale of items (i.e., sub-trajectory) to be indexed.

To the best of our knowledge, this is the only work to consider the fusion of topic model and activity trajectory search (based on spatial and thematic meaning domains), so that activity trajectories can be recommended more rationally by the interpretation of activity descriptions and user intentions.

7 Conclusions

This paper studies the problem of searching activity trajectories more effectively by replacing keyword matching with semantic interpretation. The probabilistic topic model is used to interpret the thematic meaning of keywords associated in trajectories and user queries. To support the efficient search for the activity trajectory that has spatial and thematic proximity to the user query, we developed a novel hybrid index structure called TP-tree, and proposed effective searching algorithms to prune the search space. Extensive experimental results on real datasets demonstrated the effectiveness of our proposed method.

In the future, it will be interesting to investigate how to incorporate the frequent human mobility and activity patterns into the system, to recommend users more useful and informative activity trajectories.

References

[1] Xiao X, Zheng Y, Luo Q, Xie X. Finding similar users using category-based location history. In *Proc. the 18th ACM SIGSPATIAL International Symposium on Advances in Geographic Information Systems*, November 2010, pp.442-445.

[2] Zheng Y, Xie X. Learning location correlation from GPS trajectories. In *Proc. the 11th Int. Conference on Mobile Data Management*, May 2010, pp.27-32.

[3] Cao X, Cong G, Jensen C S. Mining significant semantic locations from GPS data. *Proceedings of the VLDB Endowment*, 2010, 3(1): 1009-1020.

[4] Zheng Y, Zhang L, Xie X, Ma W Y. Mining interesting locations and travel sequences from GPS trajectories. In *Proc. the 18th Int. Conference on World Wide Web*, April 2009, pp.791-800.

[5] Chen Z, Shen H T, Zhou X, Zheng Y, Xie X. Searching trajectories by locations: An efficiency study. In *Proc. the 2010 ACM SIGMOD Int. Conference on Management of Data*, June 2010, pp.255-266.

[6] Xu J, Gao Y, Liu C, Zhao L, Ding Z. Efficient route search on hierarchical dynamic road networks. *Distributed and Parallel Databases*, 2015, 33(2): 227-252.

[7] Dai J, Liu C, Xu J, Ding Z. On personalized and sequenced route planning. *World Wide Web: Internet and Web Information Systems*, 2016, 19(4): 679-705.

[8] Xue A Y, Zhang R, Zheng Y, Xie X, Huang J, Xu Z. Destination prediction by sub-trajectory synthesis and privacy protection against such prediction. In *Proc. the 29th Int. Conference on Data Engineering*, April 2013, pp.254-265.

[9] Zheng K, Shang S, Yuan N J, Yang Y. Towards efficient search for activity trajectories. In *Proc. the 29th Int. Conference on Data Engineering*, April 2013, pp.230-241.

[10] Liu H, Xu J, Zheng K, Liu C, Du L, Wu X. Semantic-aware query processing for activity trajectories. In *Proc. the 10th ACM Int. Conference on Web Search and Data Mining*, February 2017, pp.283-292.

[11] Blei D M, Ng A Y, Jordan M I. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 2003, 3: 993-1022.

[12] Jagadish H V, Ooi B C, Tan K L, Yu C, Zhang R. iDistance: An adaptive B^+-tree based indexing method for nearest neighbor search. *ACM Transactions on Database Systems*, 2005, 30(2): 364-397.

[13] Blei D M. Probabilistic topic models. *Communications of the ACM*, 2012, 55(4): 77-84.

[14] Li J, Liu C, Yu J X, Chen Y, Sellis T, Culpepper J S. Personalized influential topic search via social network summarization. *IEEE Transactions on Knowledge and Data Engineering*, 2016, 28(7): 1820-1834.

[15] Blei D M, Lafferty J D. Dynamic topic models. In *Proc. the 23rd Int. Conference on Machine Learning*, June 2006, pp.113-120.

[16] Kim S, Smyth P. Hierarchical Dirichlet processes with random effects. In *Proc. the 20th Annual Conference on Neural Information Processing Systems*, December 2007, pp.697-704.

[17] Du L, Buntine W L, Jin H. Sequential latent Dirichlet allocation: Discover underlying topic structures within a document. In *Proc. the 10th IEEE International Conference on Data Mining*, December 2010, pp.148-157.

[18] Jiang H, Zhou R, Zhang L, Wang H, Zhang Y. A topic model based on Poisson decomposition. In *Proc. the 2017 ACM Conference on Information and Knowledge Management*, November 2017, pp.1489-1498.

[19] Li B, Yang X, Zhou R, Wang B, Liu C, Zhang Y. An efficient method for high quality and cohesive topical phrase mining. *IEEE Transactions on Knowledge and Data Engineering*, 2019, 31(1): 120-137.

[20] Li B, Yang X, Zhou R, Wang B, Liu C, Zhang Y. Sentence level topic models for associated topics extraction. *World Wide Web*: *Internet and Web Information Systems: Special Issue on Web and Big Data*, 2018, Article No. 7.

[21] Liu Q, Ge Y, Li Z, Chen E, Xiong H. Personalized travel package recommendation. In *Proc. the 11th IEEE Int. Conference on Data Mining*, December 2011, pp.407-416.

[22] Hu B, Jamali M, Ester M. Spatio-temporal topic modeling in mobile social media for location recommendation. In *Proc. the 13th IEEE Int. Conference on Data Mining*, December 2013, pp.1073-1078.

[23] Yuan N J, Zheng Y, Xie X, Wang Y, Zheng K, Xiong H. Discovering urban functional zones using latent activity trajectories. *IEEE Trans. Knowledge and Data Engineering*, 2015, 27(3): 712-725.

[24] Cong G, Jensen C S, Wu D. Efficient retrieval of the top-k most relevant spatial web objects. *Proceedings of the VLDB Endowment*, 2009, 2(1): 337-348.

[25] Rocha-Junior J B, Gkorgkas O, Jonassen S, Nørvåg K. Efficient processing of top-k spatial keyword queries. In *Proc. the 12th International Symposium on Spatial and Temporal Databases*, August 2011, pp.205-222.

[26] Zhang D, Chan C Y, Tan K L. Processing spatial keyword query as a top-k aggregation query. In *Proc. the 37th Int. ACM SIGIR Conference on Research and Development in Information Retrieval*, July 2014, pp.355-364.

[27] de Felipe I, Hristidis V, Rishe N. Keyword search on spatial databases. In *Proc. the 24th Int. Conference on Data Engineering*, April 2008, pp.656-665.

[28] Tao Y, Sheng C. Fast nearest neighbor search with keywords. *IEEE Trans. Knowledge and Data Engineering*, 2014, 26(4): 878-888.

[29] Chen Y Y, Suel T, Markowetz A. Efficient query processing in geographic Web search engines. In *Proc. the 2006 ACM SIGMOD International Conference on Management of Data*, June 2006, pp.277-288.

[30] Zhang C, Zhang Y, Zhang W, Lin X, Cheema M A, Wang X. Diversified spatial keyword search on road networks. In *Proc. the 17th International Conference on Extending Database Technology*, March 2014, pp.367-378.

[31] Gao Y, Qin X, Zheng B, Chen G. Efficient reverse top-k Boolean spatial keyword queries on road networks. *IEEE Trans. Knowledge and Data Engineering*, 2015, 27(5): 1205-1218.

[32] Luo S, Luo Y, Zhou S, Cong G, Guan J, Yong Z. Distributed spatial keyword querying on road networks. In *Proc. the 17th International Conference on Extending Database Technology*, March 2014, pp.235-246.

[33] Zheng K, Zheng B, Xu J, Liu G, Liu A, Li Z. Popularity-aware spatial keyword search on activity trajectories. *World Wide Web*: *Internet and Web Information Systems*, 2017, 20(4): 749-773.

[34] Cao X, Cong G, Jensen C S, Ooi B C. Collective spatial keyword querying. In *Proc. the 2011 ACM SIGMOD International Conference on Management of Data*, June 2011, pp.373-384.

[35] Chen Q, Hu H, Xu J. Authenticating top-k queries in location-based services with confidentiality. *Proceedings of the VLDB Endowment*, 2013, 7(1): 49-60.

[36] Li J, Liu C, Islam M S. Keyword-based correlated network computation over large social media. In *Proc. the 30th IEEE International Conference on Data Engineering*, March 2014, pp.268-279.

[37] Wu D, Choi B, Xu J, Jensen C S. Authentication of moving top-k spatial keyword queries. *IEEE Transactions on Knowledge and Data Engineering*, 2015, 27(4): 922-935.

[38] Guo L, Shao J, Aung H H, Tan K L. Efficient continuous top-k spatial keyword queries on road networks. *GeoInformatica*, 2015, 19(1): 29-60.

[39] Qian Z, Xu J, Zheng K, Sun W, Li Z, Guo H. On efficient spatial keyword querying with semantics. In *Proc. the 21st International Conference on Database Systems for Advanced Applications*, April 2016, pp.149-164.

[40] Qian Z, Xu J, Zheng K, Zhao P, Zhou X. Semantic-aware top-k spatial keyword queries. *World Wide Web*: *Internet and Web Information Systems*, 2018, 21(3): 573-594.

[41] Zheng Y, Liu Y, Yuan J, Xie X. Urban computing with taxicabs. In *Proc. the 13th Int. Conference on Ubiquitous Computing*, September 2011, pp.89-98.

[42] Xie M. EDS: A segment-based distance measure for sub-trajectory similarity search. In *Proc. the 2014 ACM SIGMOD International Conference on Management of Data*, June 2014, pp.1609-1610.

[43] Su H, Zheng K, Wang H, Huang J, Zhou X. Calibrating trajectory data for similarity-based analysis. In *Proc. the 2013 ACM SIGMOD International Conference on Management of Data*, June 2013, pp.833-844.

[44] Xie X, Yiu M L, Cheng R, Lu H. Scalable evaluation of trajectory queries over imprecise location data. *IEEE Transactions on Knowledge and Data Engineering*, 2014, 26(8): 2029-2044.

[45] Jiang W, Zhu J, Xu J, Li Z, Zhao P, Zhao L. A feature based method for trajectory dataset segmentation and profiling. *World Wide Web*: *Internet and Web Information Systems*, 2017, 20(1): 5-22.

[46] Bogorny V, Kuijpers B, Alvares L O. ST-DMQL: A semantic trajectory data mining query language. *International Journal of Geographical Information Science*, 2009, 23(10): 1245-1276.

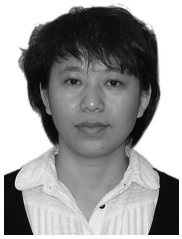
[47] Alvares L O, Bogorny V, Kuijpers B, de Macêdo J A F, Moelans B, Vaisman A. A model for enriching trajectories with semantic geographical information. In *Proc. the 15th ACM International Symposium on Geographic Information Systems*, November 2007, Article No. 22.

[48] Ying J J C, Lee W C, Weng T C, Tseng V S. Semantic trajectory mining for location prediction. In *Proc. the 19th ACM SIGSPATIAL International Symposium on Advances in Geographic Information Systems*, November 2011, pp.34-43.

[49] Leung K W T, Lee D L, Lee W C. CLR: A collaborative location recommendation framework based on co-clustering. In *Proc. the 34th ACM SIGIR Conference on Research and Development in Information Retrieval*, July 2011, pp.305-314.

[50] Zheng V W, Zheng Y, Xie X, Yang Q. Collaborative location and activity recommendations with GPS history data. In *Proc. the 19th Int. Conference on World Wide Web*, April 2010, pp.1029-1038.

[51] Shang S, Ding R, Yuan B, Xie K, Zheng K, Kalnis P. User oriented trajectory search for trip recommendation. In *Proc. the 15th Int. Conference on Extending Database Technology*, March 2012, pp.156-167.



Li-Hua Yin received her Ph.D. degree in computer science and technology from Harbin Institute of Technology, Harbin, in 2007. She is a professor at Cyberspace Institute of Advanced Technology, Guangzhou University, Guangzhou. Her research interests include information security, big data privacy protection, etc. She is a member of CCF and CIPS.

Huiwen Liu received her Master's degree in the computer science and technology from Soochow University, Suzhou, in 2017. She is currently working toward her Ph.D. degree at Singapore Management University (SMU), Singapore. Her research interests mainly include data mining, artificial intelligence and blockchain.